▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 36

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

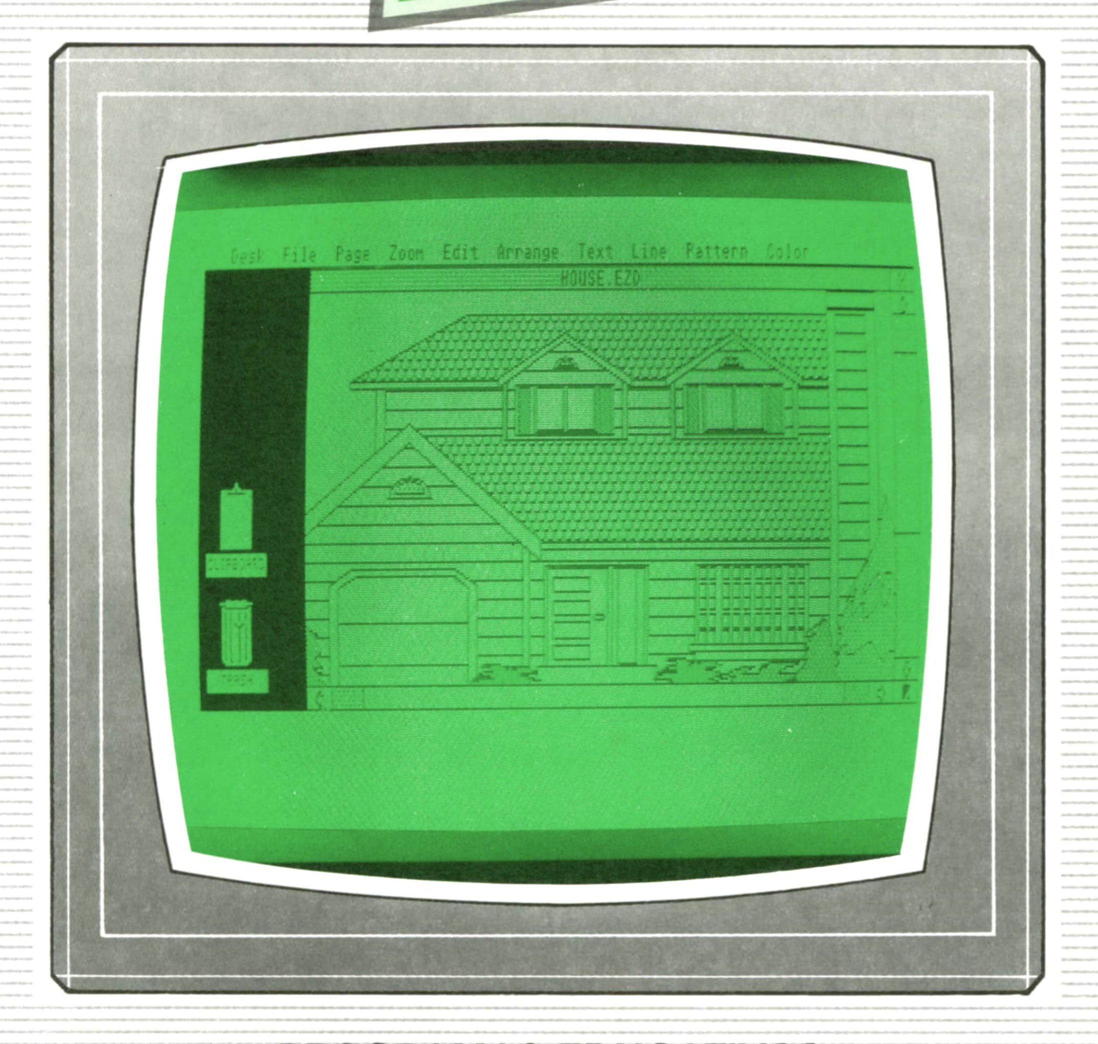
▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación

36

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Sociedad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (COBOL): Eloy Pérez, Licenciado en Informática. Ana Pastor, Licenciada en Informática.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 455 09 99. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-188-6
ISBN de la obra: 84-7688-058-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

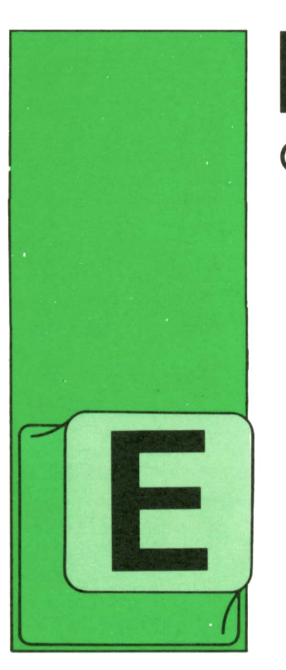
Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 259 73 31. 28020 Madrid.
Marzo, 1988
P.V.P. Canarias: 335,–.

INDICE

INFORMATICA BASICA

COMPONENTES DE UN SISTEMA DE COMUNICACIONES

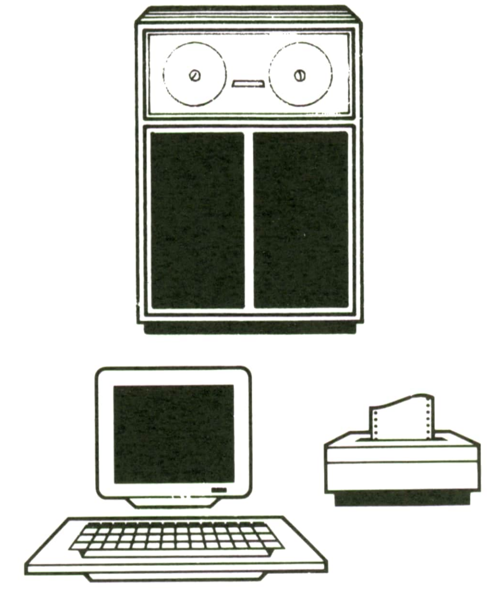
Los medios de teleinformática



EN el capítulo anterior veíamos de forma global qué era y en qué consiste la teleinformática, o tratamiento «a distancia» de la información. Para poder implantar un sistema de teleproceso son siempre necesarios una serie de elementos o subsistemas que detallaremos más adelante.



El ordenador central



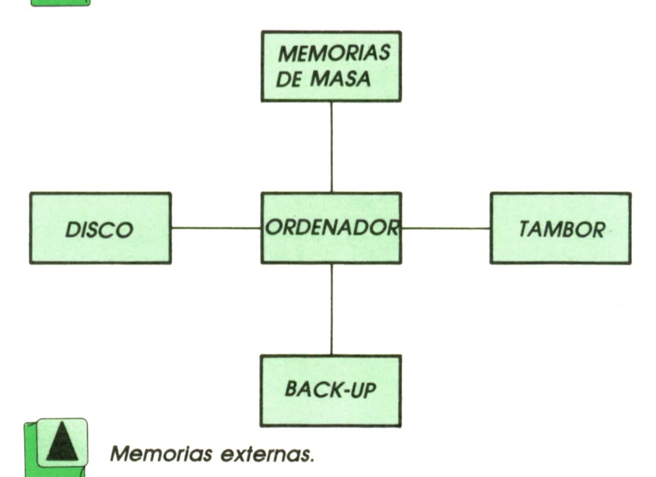


El ordenador central: el principal elemento de la teleinformática.

Ya que es el principal factor de control del sistema, deberá tener unas características hardware y software muy apropiadas al trabajo a realizar, para poder controlar de manera eficiente todo el intercambio de informaciones dentro del sistema. Estas caracteristicas, que tendrán que tenerse en cuenta a la hora de elegir un ordenador con vistas a aplicaciones telemáticas, son:

1. *Modularidad.* Importante a la hora de tener que ampliar el equipo debido a incrementos de información, ya que no será necesario un total replanteamiento de los análisis y la programación de las aplicaciones ya en funcionamiento.
2. *Buffer.* Para procesar la información que llega al centro de manera totalmente aleatoria, será necesaria la creación de una serie de colas (a la entrada, durante el proceso y a la salida), en las que estarán las informaciones pendientes de tratamiento. Para ello será necesario definir una serie de prioridades de acceso y tener disponible un pequeño sistema de almacenamiento, donde permanecería la información en espera de proceso; esto es lo que constituye el buffer.
3. *Reubicación de programas.*
4. *Protección de memoria.*
5. *Reloj.*

Memorias externas



Memorias externas.

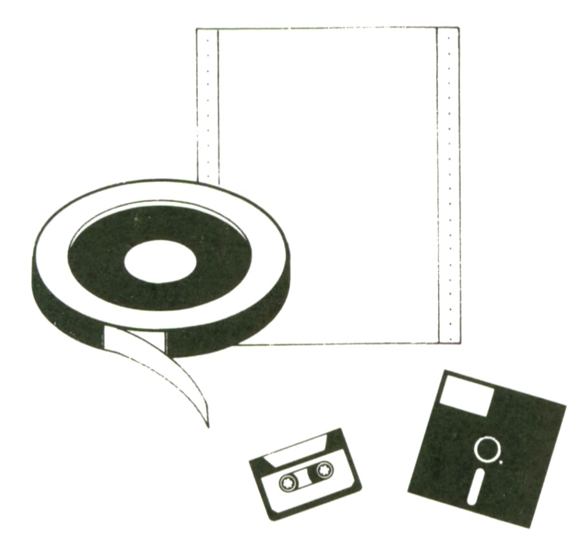
Como muestra la figura, podemos distinguir dos grandes categorías de memorias externas: las primeras, de acceso directo, contienen los ficheros tratados

desde los terminales remotos, y las segundas, de acceso secuencial, son para aplicaciones más inmediatas.



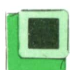
Ficheros

Tan importantes como el ordenador, por lo que la utilización de los datos a distancia llevará consigo el correcto almacenamiento de los datos en soportes magnéticos.





Diferentes soportes para almacenar información.



Periféricos

Se puede prescindir de ellos, salvo que el ordenador esté totalmente dedicado a este tipo de trabajo en exclusiva.



Canales de comunicación

Como ya dijimos, uno de los canales más empleado es el de la línea telefónica, y en casos muy particulares, otras como enlaces de radio, por cables coaxiales, enlaces vía satélite, etc.

La diferencia entre unos y otros medios, aparte del precio, se establece en la anchura de banda —como veremos más adelante—, la velocidad de transmisión y el número de comunicaciones simultáneas que admiten.

La velocidad de transmisión por una línea se mide en *baudios,* o unidades elementales de transmisión por segundo. Lo más usual es que dicha unidad elemental sea un bit, en cuyo caso un baudio será un *bit por segundo.*

Una de las características que distinguen a unas líneas de otras es si son o no completas. Mediante las líneas conmutadas la conexión ordenador-terminal no es permanente; debe restablecerse para cada mensaje o serie de mensajes transmitidos; mediante las líneas no conmutadas la conexión está establecida de forma permanente.

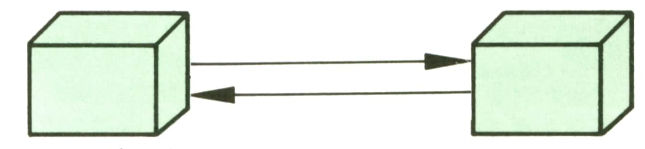
NORMAS DE COMUNICACION



Circuito Simplex, sólo puede transmitir en un sentido.



Transmisión Semi-dúplex, permite que los datos lleguen en ambos sentidos.



Transmisión dúplex, permite que los datos viajen en ambos sentidos a la vez.

En función de si los datos se puedan transmitir en uno o dos sentidos, la comunicación puede ser SIMPLEX, SEMIDUPLEX y DUPLEX.

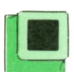
La forma de comunicación la establece el tipo de dispositivo que conforma la red y para cada tipo de comunicación se requieren tipos de líneas con determinadas características. Para una comunicación simplex y semidúplex, la línea puede ser de dos hilos; pero para comunicar en dúplex hace falta una línea de cuatro hilos.

Otra diferencia entre las líneas de comunicación es su velocidad, es decir, la cantidad de bits que puede transmitir por segundo. Las distintas velocidades pueden ser clasificadas en tres grupos.

Control de líneas

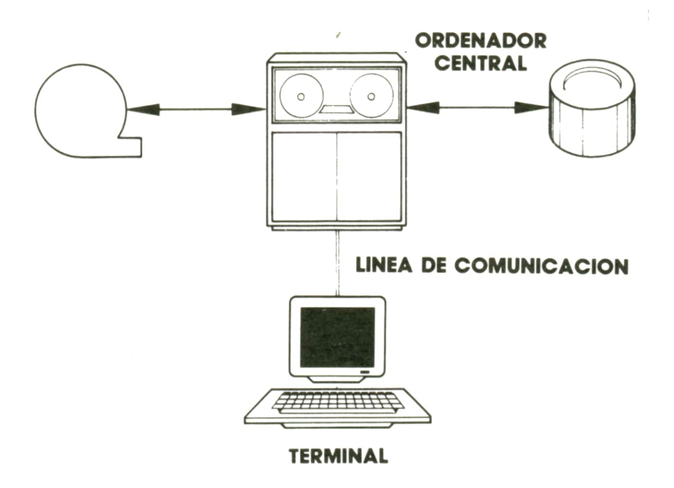
Se utiliza este elemento para el correcto funcionamiento entre el ordenador y

las líneas de comunicación. Se encarga de importantes trabajos que más adelante describiremos.



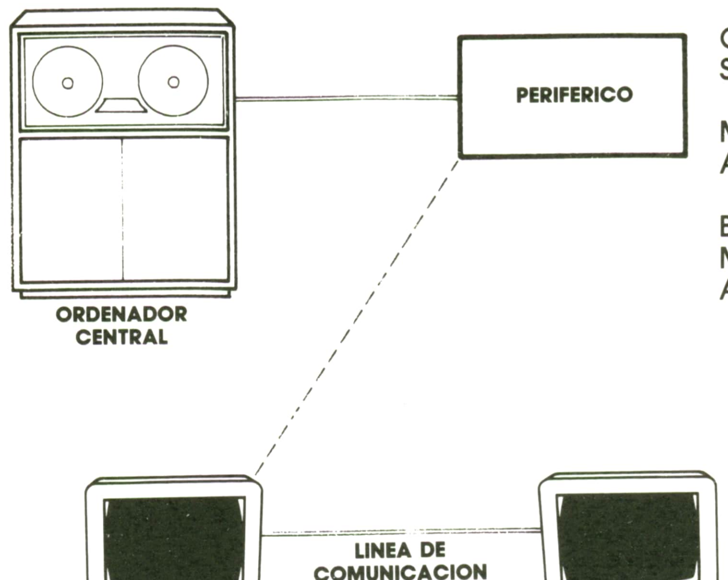
Comunicaciones

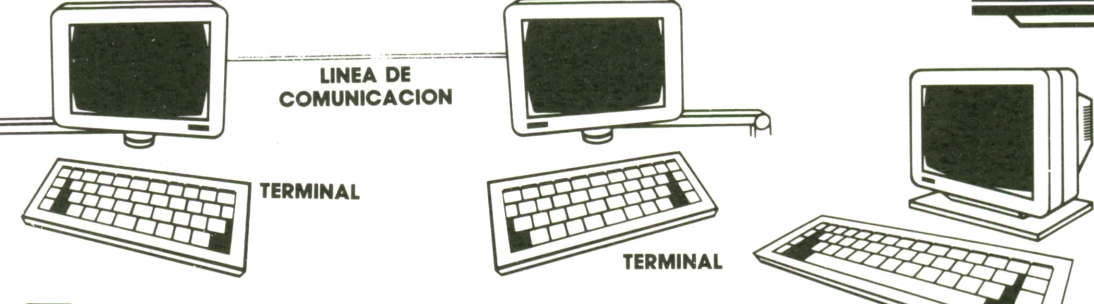
Soporte material (líneas) o inmaterial (enlaces de radio) para la circulación de información entre el ordenador y los terminales.





Sistema ON-LINE.





Sistema OFF-LINE.

Terminales

Muy importantes en los sistemas de teleinformática, ya que a través de ellos se envían y/o se reciben los datos a través de los canales de comunicación.

Son dispositivos transmisores o receptores que permiten al operador enlazar con el sistema o con otros terminales. Generalmente no tienen capacidad de cálculo. Normalmente un sistema de teletratamiento implica un ordenador central y un conjunto de terminales. Por otra parte, cualquier periférico usado normalmente puede ser manejado a distancia, siempre que se incluya los dispositivos de control necesarios.

Las principales características que definen un terminal son:

— Forma de conexión:
On-line.
Off-line.

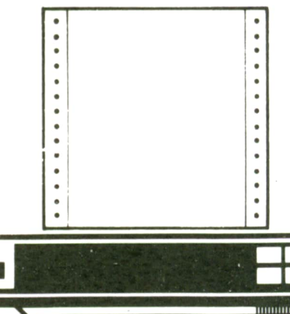
— Cometido:
Transmisión.
Recepción.
Mixto.

— Modalidad de transmisión:
Síncrona o asíncrona.
Simplex, semi-dúplex o dúplex.
En paralelo o en serie.

— Capacidad de almacenamiento:
Con buffer.
Sin buffer.

— Operación:
Manual.
Automática.

— Velocidad:
Baja.
Media.
Alta.

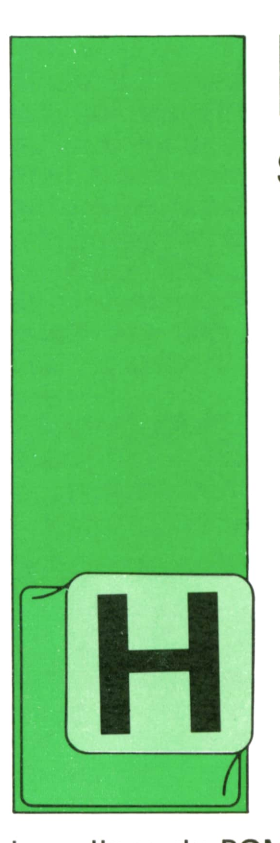




La impresora y el monitor dos tipos de terminales.

MAQUINA Z-80

SPECTRUM, AMSTRAD, MSX





Sistemas operativos

HASTA ahora habíamos considerado que no existía ningún otro programa residiendo simultáneamente al nuestro en la memoria. En la práctica esto no es así; existen las rutinas de ROM, pertenecientes al ordenador, que estemos utilizando.

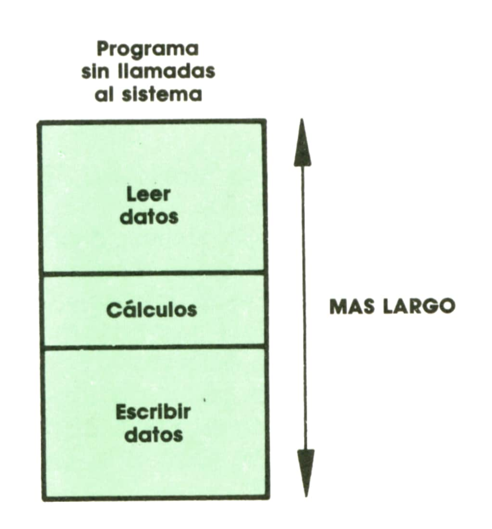
El sistema operativo de nuestro ordenador es un programa que está ejecutándose en todo momento, controlando todas las operaciones que queramos realizar. Ocasionalmente pasa el control al programa que queramos ejecutar, tales como nuestras rutinas, el intérprete de BASIC, etc. Aun en este caso, ciertas operaciones comunes, tales como la lectura de un carácter del teclado o la escritura en la pantalla, la realizan llamando, mediante instrucciones JP, a las rutinas existentes en el sistema operativo.

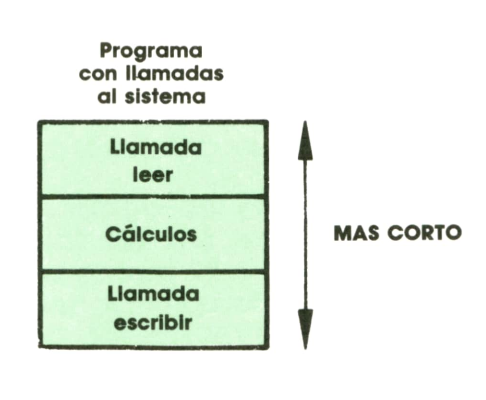
Así, al realizar rutinas en código máquina debemos aprovechar al máximo las ya existentes en la ROM de nuestro aparato. Por ello recomendamos que el usuario se haga con algún libro en el que vengan, comentadas, las rutinas de su sistema. Este es necesario porque aparte de conocer la dirección donde empieza dicha rutina, es necesario conocer qué parámetros debemos pasarlas para funcionar, qué registros son afectados, para salvarlos con anterioridad, y dónde nos deja los resultados.

Existen algunos sistemas en que se accede a las rutinas del sistema saltando ciertas posiciones. De esta forma pueden variar las versiones e implementaciones de un mismo sistema operativo, permaneciendo constantes las zonas donde están las llamadas al sistema.

En algunos ordenadores basados en el Z-80 debemos tener cuidado de que nuestras rutinas en código máquina no interfieran el funcionamiento del sistema operativo.

Esto es especialmente importante al in-





tentar usar interrupciones, ya que los resultados pueden ser desastrosos, quedándose la máquina «colgada», sin responder al teclado, siendo necesario apagarla y recomenzar de nuevo.

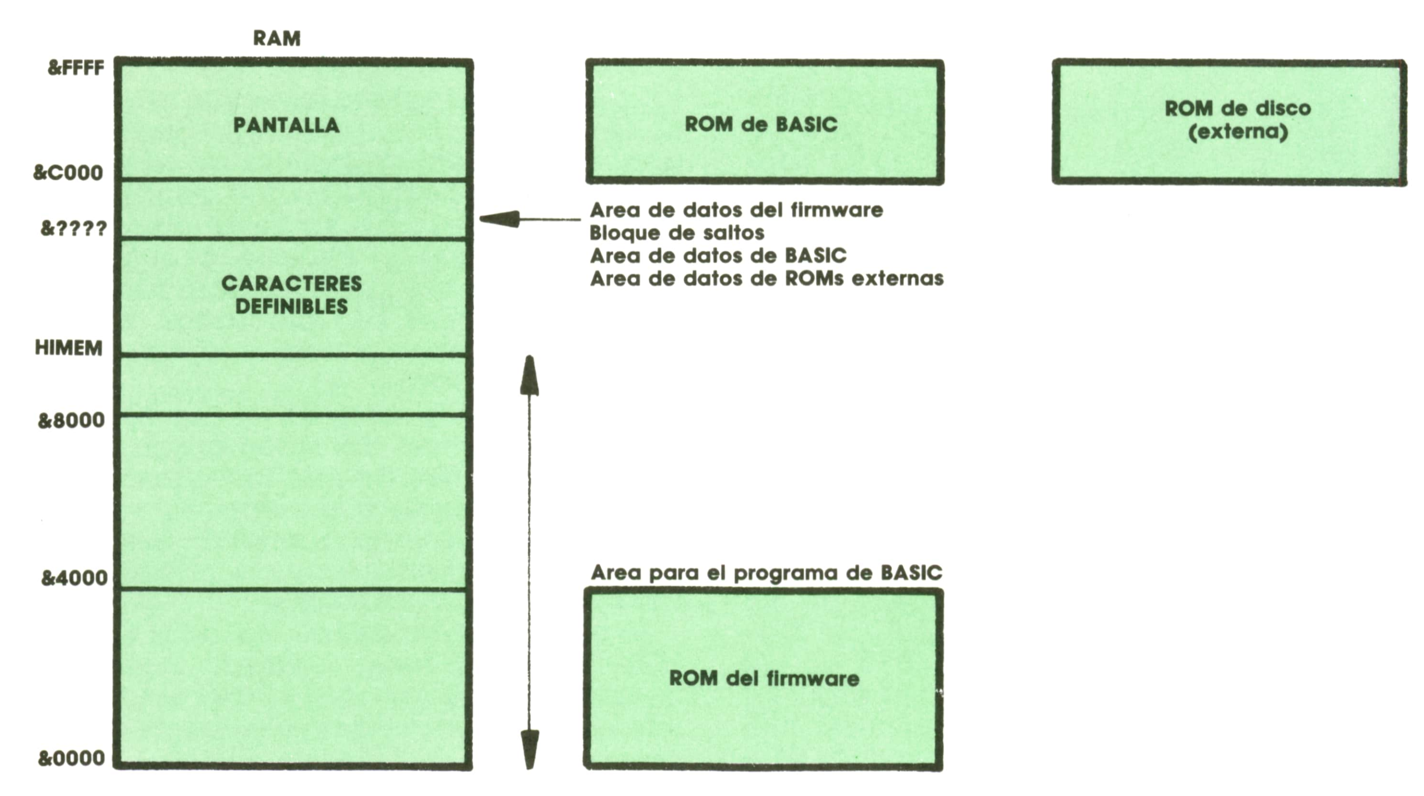
También es necesario colocarlo en una zona de memoria que no interfiera con otros programas.

Esto se complica un poco en los AMSTRAD que llevan más de 64 Kbytes de memoria.

Al no poder direccionar el Z-80 más que 64K, el resto debe añadirse mediante técnicas de paginación.

Esto consiste en dividir la memoria en diversos bloques o barras que se conmutan entre sí; es decir, disponemos diversas zonas de la RAM que tienen las mismas direcciones accediéndose a cada una según desee el usuario. Por si esto fuese poco, además, está la ROM. Esta se direcciona por la señal de búsqueda de dirección, por lo que pueden tener la misma dirección de una zona de RAM en la que almacenan datos. Por ello debe desactivarse la ROM en la zona en que esté nuestro programa.

Además, si realizamos llamadas al sistema también debemos asegurarnos que la ROM en esa zona esté activada.



La forma de realizar las llamadas es mediante la instrucción RST.

De las direcciones de entrada/salida la mayoría están reservadas por el ordenador.

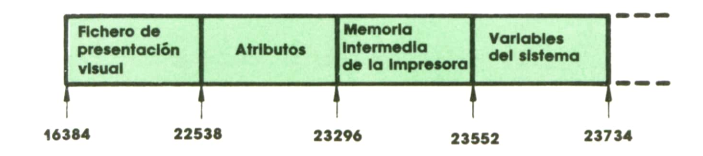
Las direcciones inferiores a $7FFF no deben utilizarse, pues colisionaría con el sistema.

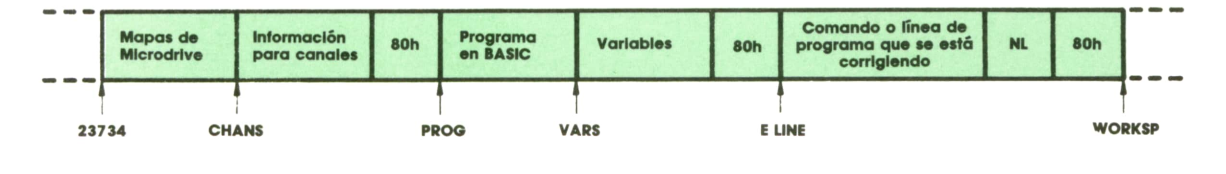
Las direcciones de los registros de los dispositivos pueden ser como $F8??, SF9?? y $FB??, donde ?? puede ser DC y FF para interfaces de comunicaciones y entre EO y FE para otros periféricos.

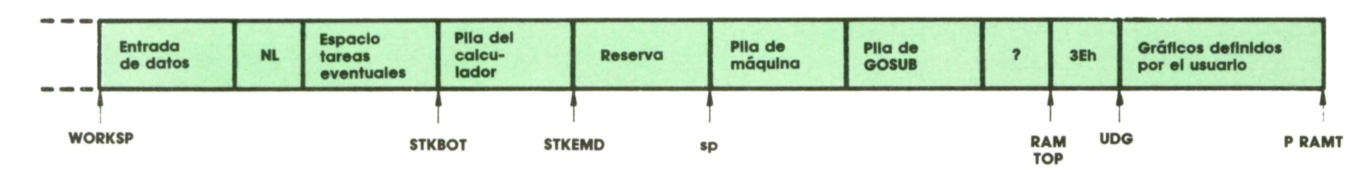
Todo esto es mucho más sencillo en el SPECTRUM. Se pueden colocar las rutinas del usuario en cualquier parte de la RAM, pero para no interferir con el funcionamiento del BASIC lo más aconsejable es colocarlo en las posiciones a partir de las indicadas por el vector RAMTOP situado en la posición 23730.

El teclado utiliza la posición $FE para leer el carácter escrito.

La misma dirección sirve como salida para controlar el altavoz. Todas las demás están libres, pero debemos tener cuidado con la ULA si conectamos algo al SPECTRUM, ya que éste realiza interrupciones cuando lo cree oportuno, por lo que podría estropearse. Para llamar a rutinas de BASIC dar un salto a la dirección de comienzo de ésta.





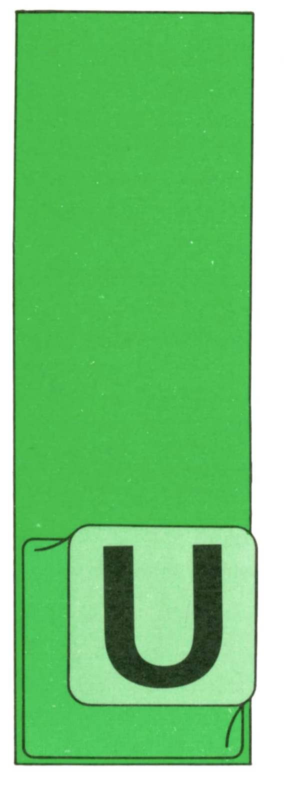


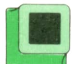
Existen otros ordenadores que utilizan el Z-80 como procesador principal. La mayoría de estos sistemas ejecutan el sistema operativo CP/M. En éste las direcciones donde están las utilidades predefinidas para el uso del sistema operativo permanecen constantes, llamadas al BIOS (BASIC Input Output System = Sistema básico de entrada salida), que gestiona todas las operaciones de más bajo nivel y el BFMS 'BASIC File Managements System = Sistema básico de gestión de ficheros), que controla los ficheros de disco y los ficheros asignados a distintos periféricos (teclado, pantalla CRT, etc.).

Por todo ello, los programas en código máquina del Z-80 no pueden realizarse de espaldas al sistema operativo que estemos utilizando, sino en perfecta conexión con él.

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS

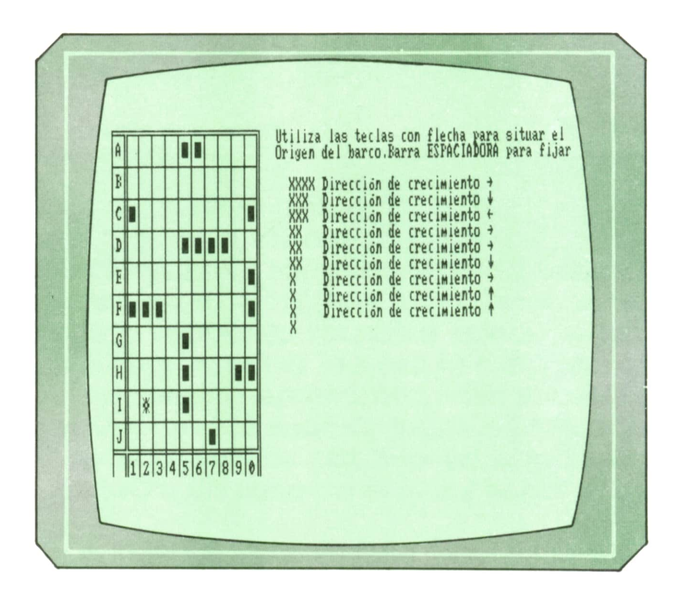




Barquitos

UNO de los juegos más utilizados por todos los estudiantes de todo el mundo, sean del curso que sean, es el de los barquitos.

Aún así, es deficil encontrar un programa que nos permita jugar contra el ordenador y que esté correctamente escrito y nos pueda entretener.



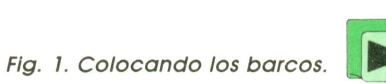


Fig. 1. Colocando los barcos.

```
PROGRAMA: GUERRA DE BARCOS
==========================

1000 REM **********************************************************************
1010 REM ******        JUEGO DE LOS BARQUITOS (CONTRA EL ORDENADOR)        ******
1020 REM ******                                                            ******
1030 REM ******        POR : JUAN CARLOS GONZALEZ YAGE                    ******
1040 REM **********************************************************************
1050 REM
1060 REM **********************************************************************
1070 REM **************    (c) EDICIONES SIGLO CULTURAL, 1987    *****************
1080 REM **********************************************************************
1090 REM
1100 REM ****************************
1110 REM * INICIALIZACION DEL JUEGO *
1120 REM ****************************
1130 REM
1140 CLS:KEY OFF:DIM PTO(100,100)
1150 REM
1160 REM *** CREACION PRIMERA PANTALLA ***
1170 REM
1180 GOSUB 1750
1190 REM
1200 REM *** UBICACION DE LA FLOTA DEL JUGADOR ***
1210 REM
1220 GOSUB 2230
```

```
1230 REM
1240 REM *** CREACION SEGUNDA PANTALLA ***
1250 REM
1260 FOR C=1 TO 15
1270    LOCATE C,30:PRINT SPACE$(50)
1280 NEXT C
1290 GOSUB 3850
1300 REM
1310 REM *** EL ORDENADOR SITUA SUS BARCOS ***
1320 REM
1330 GOSUB 4470
1340 REM
1350 REM *** SE COMPLETA LA PANTALLA DE JUEGO ***
1360 REM
1370 GOSUB 4260
1380 REM
1390 REM ***********************************
1400 REM *   F A S E    D E    J U E G O   *
1410 REM ***********************************
1420 REM
1430 IF BUNO=10 THEN GOTO 1570
1440 GOSUB 5180
1450 IF RPTIRJ=1 THEN LET RPTIRJ=0:GOTO 1430
1460 REM
1470 REM
1480 REM *** DISPARA EL ORDENADOR ***
1490 REM
1500 IF BUNJ=10 THEN GOTO 1660
1510 GOSUB 6220
1520 IF RPTIRO=1 THEN LET RPTIRO=0 :GOTO 1500
1530 GOTO 1430
1540 REM
1550 REM ***  G A N A   E L   J U G A D O R  ***
1560 REM
1570 FOR C=1 TO 24
1580    LOCATE C,29:PRINT SPACE$(23)
1590 NEXT C
1600 LOCATE 8,35:PRINT"FELICIDADES"
1610 LOCATE 10,34:PRINT"ME HAS GANADO"
1620 GOTO 1710
1630 REM
1640 REM *** GANA EL ORDENADOR ***
1650 REM
1660 FOR C=1 TO 24
1670    LOCATE C,29:PRINT SPACE$(23)
1680 NEXT C
1690 LOCATE 8,37:PRINT"LO SIENTO"
1700 LOCATE 10,35:PRINT"HAS PERDIDO"
1710 LOCATE 23,33:PRINT"Pulsa un tecla"
1720 A$=INKEY$:IF A$="" THEN 1720
1730 CLS
1740 END
1750 REM
1760 REM *** SUBRUTINA PRINCIPAL DE PRIMERA PANTALLA ***
1770 REM
1780 DES=10:HAS=26:CAR=209:CX=2
1790 GOSUB 2190
1800 FOR CX=4 TO 20 STEP 2
1810    DES=7:HAS=27:CAR=196
1820    GOSUB 2190
1830    DES=10:HAS=26:CAR=197
1840    GOSUB 2190
1850    LOCATE CX,6:PRINT CHR$(195)
1860    LOCATE CX,8:PRINT CHR$(215)
1870    LOCATE CX,28:PRINT CHR$(182)
1880 NEXT CX
```

```
1890 FOR CX=3 TO 23 STEP 2
1900     DES=6:HAS=26:CAR=179
1910     GOSUB 2190
1920     LOCATE CX,8:PRINT CHR$(186)
1930     LOCATE CX,28:PRINT CHR$(186)
1940 NEXT CX
1950 DES=7:HAS=27:CX=2:CAR=205
1960 GOSUB 2190
1970 CX=22
1980 GOSUB 2190
1990 DES=10:HAS=26:CX=22:CAR=216
2000 GOSUB 2190
2010 LOCATE 2,6:PRINT CHR$(213)
2020 LOCATE 2,8:PRINT CHR$(203)
2030 LOCATE 2,28:PRINT CHR$(187)
2040 LOCATE 22,6:PRINT CHR$(198)
2050 LOCATE 22,8:PRINT CHR$(206)
2060 LOCATE 22,28:PRINT CHR$(185)
2070 CTL=64
2080 FOR CX=3 TO 21 STEP 2
2090    CTL=CTL+1
2100    LOCATE CX,7:PRINT CHR$(CTL)
2110 NEXT CX
2120 CTN=48
2130 FOR CY=9 TO 25 STEP 2
2140     CTN=CTN+1
2150     LOCATE 23,CY:PRINT CHR$(CTN)
2160 NEXT CY
2170 LOCATE 23,27:PRINT CHR$(48)
2180 RETURN
2190 FOR COL=DES TO HAS STEP 2
2200     LOCATE CX,COL:PRINT CHR$(CAR)
2210 NEXT COL
2220 RETURN
2230 REM
2240 REM *** SUBRUTINA PARA SITUAR LA FLOTA ***
2250 REM
2260 LOCATE 2,33:PRINT"Utiliza las teclas con flecha para situar el"
2270 LOCATE 3,33:PRINT"Origen del barco.Barra ESPACIADORA para fijar"
2280 LGB=4:CB=5
2290 GOSUB 2490
2300 LGB=3:CB=6
2310 GOSUB 2490
2320 LGB=3:CB=7
2330 GOSUB 2490
2340 LGB=2:CB=8
2350 GOSUB 2490
2360 LGB=2:CB=9
2370 GOSUB 2490
2380 LGB=2:CB=10
2390 GOSUB 2490
2400 LGB=1:CB=11
2410 GOSUB 2490
2420 LGB=1:CB=12
2430 GOSUB 2490
2440 LGB=1:CB=13
2450 GOSUB 2490
2460 LGB=1:CB=14
2470 GOSUB 2490
2480 RETURN
2490 CODCB$=STRING$(LGB,88)
2500 LOCATE CB,35:PRINT CODCB$
2510 GOSUB 2770
2520 LOCATE CB,40:PRINT"Direcci"n de crecimiento"
```

```
2530 COLOR 18
2540 LOCATE CB,65:PRINT CHR$(22)
2550 COLOR 2
2560 A$=INKEY$:IF A$="" THEN 2560
2570 IF A$="8" THEN DCTO=1:Z=24:GOTO 2620
2580 IF A$="4" THEN DCTO=2:Z=27:GOTO 2620
2590 IF A$="2" THEN DCTO=3:Z=25:GOTO 2620
2600 IF A$="6" THEN DCTO=4:Z=26:GOTO 2620
2610 BEEP:GOTO 2560
2620 LOCATE CB,65:PRINT CHR$(Z)
2630 IF DCTO=1 THEN GOTO 3150
2640 IF DCTO=2 THEN GOTO 3320
2650 IF DCTO=3 THEN GOTO 3490
2660 GOTO 3660
2670 IF ERCTO=1 THEN ERCTO=0:GOTO 2690
2680 RETURN
2690 LOCATE CB,40:PRINT"Imposible Situar el Barco Aqu!"
2700 LOCATE CB+1,47:PRINT"(Pulsa una tecla)
2710 BEEP
2720 A$=INKEY$:IF A$="" THEN 2720
2730 LOCATE CB,40:PRINT SPACE$(30)
2740 LOCATE CB+1,47:PRINT SPACE$(17)
2750 LOCATE CX,CY:PRINT CHR$(32)
2760 GOTO 2490
2770 CX=13:CY=17
2780 COLOR 18
2790 LOCATE CX,CY:PRINT CHR$(15)
2800 COLOR 2
2810 A$=INKEY$:IF A$="" THEN 2810
2820 IF A$="8" THEN 2880
2830 IF A$="4" THEN 2940
2840 IF A$="2" THEN 3000
2850 IF A$="6" THEN 3060
2860 IF ASC(A$)=32 THEN 3120
2870 BEEP:GOTO 2780
2880 CX=CX-2
2890 IF CX<3 THEN CX=CX+2:GOTO 2870
2900 IF PTO(CX+2,CY)>4 THEN Z=219:GOTO 2920
2910 Z=32
2920 LOCATE CX+2,CY:PRINT CHR$(Z)
2930 GOTO 2780
2940 CY=CY-2
2950 IF CY<9 THEN CY=CY+2:GOTO 2870
2960 IF PTO(CX,CY+2)>4 THEN Z=219:GOTO 2980
2970 Z=32
2980 LOCATE CX,CY+2:PRINT CHR$(Z)
2990 GOTO 2780
3000 CX=CX+2
3010 IF CX>21 THEN CX=CX-2:GOTO 2870
3020 IF PTO(CX-2,CY)>4 THEN Z=219:GOTO 3040
3030 Z=32
3040 LOCATE CX-2,CY:PRINT CHR$(Z)
3050 GOTO 2780
3060 CY=CY+2
3070 IF CY>27 THEN CY=CY-2:GOTO 2870
3080 IF PTO(CX,CY-2)>4 THEN Z=219:GOTO 3100
3090 Z=32
3100 LOCATE CX,CY-2:PRINT CHR$(Z)
3110 GOTO 2780
3120 IF PTO(CX,CY)>3 THEN 2870
3130 LOCATE CX,CY:PRINT CHR$(219)
3140 RETURN
3150 IF (CX-LGB*2+2)<3 THEN 3830
3160 FOR X=CX TO (CX-LGB*2+2) STEP -2
3170    IF PTO(X,CY)>3 THEN 3830
3180 NEXT X
```

```
3190 FOR N=CX TO (CX-LGB*2+2) STEP -2
3200    PTO(N,CY)=CB
3210    PTO(N,CY-2)=4
3220    PTO(N,CY+2)=4
3230    LOCATE N,CY:PRINT CHR$(219)
3240    PTO(N-2,CY-2)=4
3250    PTO(N-2,CY+2)=4
3260 NEXT N
3270 PTO(CX+2,CY-2)=4
3280 PTO(CX+2,CY)=4
3290 PTO(CX+2,CY+2)=4
3300 PTO(CX-LGB*2,CY)=4
3310 GOTO 2670
3320 IF (CY-LGB*2+2)<9 THEN 3830
3330 FOR Y=CY TO (CY-LGB*2+2) STEP -2
3340    IF PTO (CX,Y)>3 THEN 3830
3350 NEXT Y
3360 FOR N=CY TO (CY-LGB*2+2) STEP -2
3370    PTO(CX,N)=CB
3380    PTO(CX-2,N)=4
3390    PTO(CX+2,N)=4
3400    PTO(CX-2,N-2)=4
3410    PTO(CX+2,N-2)=4
3420    LOCATE CX,N:PRINT CHR$(219)
3430 NEXT N
3440 PTO(CX+2,CY+2)=4
3450 PTO(CX,CY+2)=4
3460 PTO(CX-2,CY+2)=4
3470 PTO(CX,CY-LGB*2)=4
3480 GOTO 2670
3490 IF (CX+LGB*2-2)>21 THEN 3830
3500 FOR X=CX TO (CX+LGB*2-2) STEP 2
3510    IF PTO(X,CY)>3 THEN 3830
3520 NEXT X
3530 FOR N=CX TO (CX+LGB*2-2) STEP 2
3540    PTO (N,CY)=CB
3550    PTO (N,CY-2)=4
3560    PTO (N,CY+2)=4
3570    PTO (N+2,CY-2)=4
3580    PTO (N+2,CY+2)=4
3590    LOCATE N,CY:PRINT CHR$(219)
3600 NEXT N
3610 PTO(CX-2,CY-2)=4
3620 PTO(CX-2,CY)=4
3630 PTO(CX-2,CY+2)=4
3640 PTO(CX+LGB*2,CY)=4
3650 GOTO 2670
3660 IF (CY+LGB*2-2)>27 THEN 3830
3670 FOR Y=CY TO (CY+LGB*2-2) STEP 2
3680    IF PTO(CX,Y)>3 THEN 3830
3690 NEXT Y
3700 FOR N=CY TO (CY+LGB*2-2) STEP 2
3710    PTO (CX,N)=CB
3720    PTO (CX-2,N)=4
3730    PTO (CX+2,N)=4
3740    PTO (CX-2,N+2)=4
3750    PTO (CX+2,N+2)=4
3760    LOCATE CX,N:PRINT CHR$(219)
3770 NEXT N
3780 PTO(CX-2,CY-2)=4
3790 PTO(CX,CY-2)=4
3800 PTO(CX+2,CY-2)=4
3810 PTO(CX,CY+LGB*2)=4
3820 GOTO 2670
3830 BEEP:ERCTO=1
3840 GOTO 2670
```

```
3850 DES=54:HAS=70:CX=2:CAR=209
3860 GOSUB 2190
3870 FOR CX=4 TO 20 STEP 2
3880    DES=53:HAS=73:CAR=196
3890    GOSUB 2190
3900    DES=54:HAS=70:CAR=197
3910    GOSUB 2190
3920    LOCATE CX,74:PRINT CHR$(180)
3930    LOCATE CX,72:PRINT CHR$(215)
3940    LOCATE CX,52:PRINT CHR$(199)
3950 NEXT CX
3960 FOR CX=3 TO 23 STEP 2
3970    DES=54:HAS=74:CAR= 179
3980    GOSUB 2190
3990    LOCATE CX,52:PRINT CHR$(186)
4000    LOCATE CX,72:PRINT CHR$(186)
4010 NEXT CX
4020 DES=53:HAS=73:CAR=205:CX=2
4030 GOSUB 2190
4040 CX=22
4050 GOSUB 2190
4060 DES=54:HAS=70:CX=22:CAR=216
4070 GOSUB 2190
4080 LOCATE 2,52:PRINT CHR$(201)
4090 LOCATE 2,72:PRINT CHR$(203)
4100 LOCATE 2,74:PRINT CHR$(184)
4110 LOCATE 22,52:PRINT CHR$(204)
4120 LOCATE 22,72:PRINT CHR$(206)
4130 LOCATE 22,74:PRINT CHR$(181)
4140 CTL=64
4150 FOR CX=3 TO 21 STEP 2
4160    CTL=CTL+1
4170    LOCATE CX,73:PRINT CHR$(CTL)
4180 NEXT CX
4190 CTN=48
4200 FOR CY=53 TO 71 STEP 2
4210    CTN=CTN+1
4220    LOCATE 23,CY:PRINT CHR$(CTN)
4230 NEXT CY
4240 LOCATE 23,71:PRINT CHR$(48)
4250 RETURN
4260 LOCATE 1,36 :PRINT"ORDENADOR"
4270 LOCATE 1,13:PRINT"JUGADOR"
4280 LOCATE 1,60:PRINT"ORDENADOR"
4290 LOCATE 3,35:PRINT"DISPAROS"
4300 LOCATE 5,33:PRINT"B. HUNDIDOS"
4310 LOCATE 6,35:PRINT"Al enemigo"
4320 LOCATE 7,33:PRINT"MI DISPARO"
4330 LOCATE 9,34:PRINT"RESULTADO"
4340 LOCATE 10,38:PRINT"A/T/H"
4350 FOR COL=29 TO 51
4360    LOCATE 12,COL:PRINT CHR$(206)
4370 NEXT COL
4380 LOCATE 13,37:PRINT"JUGADOR"
4390 LOCATE 15,35:PRINT"DISPAROS"
4400 LOCATE 17,33:PRINT"B. HUNDIDOS"
4410 LOCATE 18,35:PRINT"Al enemigo"
4420 LOCATE 21,36:PRINT"RESULTADO"
4430 LOCATE 23,34:PRINT"Pulsa una tecla"
4440 A$=INKEY$:IF A$="" THEN 4440
4450 LOCATE 23,34:PRINT SPACE$(15)
4460 RETURN
4470 LOCATE 3,31:PRINT"ME ESTOY SITUANDO MI"
4480 LOCATE 5,38:PRINT"FLOTA"
4490 LOCATE 7,33:PRINT"ESPERA UN MOMENTO"
4500 LGBO=4:CBO=5
4510 GOSUB 4750
```

```
4520 LGBO=3:CBO=6
4530 GOSUB 4750
4540 LGBO=3:CBO=7
4550 GOSUB 4750
4560 LGBO=2:CBO=8
4570 GOSUB 4750
4580 LGBO=2:CBO=9
4590 GOSUB 4750
4600 LGBO=2:CBO=10
4610 GOSUB 4750
4620 LGBO=1:CBO=11
4630 GOSUB 4750
4640 LGBO=1:CBO=12
4650 GOSUB 4750
4660 LGBO=1:CBO=13
4670 GOSUB 4750
4680 LGBO=1:CBO=14
4690 GOSUB 4750
4700 LOCATE 3,31:PRINT SPACE$(20)
4710 LOCATE 5,38:PRINT SPACE$(5)
4720 LOCATE 7,33:PRINT SPACE$(17)
4730 SOUND 500,3:SOUND 700,3:SOUND 400,3:SOUND 600,3
4740 RETURN
4750 ERV=0:ERH=0
4760 RANDOMIZE TIMER
4770 CX=INT(RND*18)+3
4780 IF CX MOD 2=0 THEN 4770
4790 CY=INT(RND*18)+53
4800 IF CY MOD 2=0 THEN 4790
4810 DCO=INT(RND*1)+1
4820 IF ERV=1 AND ERH=1 THEN 4750
4830 ON DCO GOTO 4840,5010
4840 IF ERV=1 THEN DCO=2:GOTO 4820
4850 IF (CX-LGBO*2+2)<3 THEN ERV=1:GOTO 4840
4860 FOR N=CX TO (CX-LGBO*2+2) STEP -2
4870     IF PTO(N,CY)>3 THEN ERV=1:GOTO 4840
4880 NEXT N
4890 FOR N=CX TO (CX-LGBO*2+2) STEP -2
4900     PTO(N,CY)=CBO
4910     PTO(N,CY-2)=4
4920     PTO(N,CY+2)=4
4930     PTO(N-2,CY-2)=4
4940     PTO(N-2,CY+2)=4
4950 NEXT N
4960 PTO(CX+2,CY-2)=4
4970 PTO(CX+2,CY)=4
4980 PTO(CX+2,CY+2)=4
4990 PTO(CX-LGBO*2,CY)=4
5000 RETURN
5010 IF ERH=1 THEN DCO=1:GOTO 4820
5020 IF(CY-LGBO*2+2)<53 THEN ERH=1:GOTO 5010
5030 FOR N=CY TO (CY-LGBO*2+2) STEP -2
5040     IF PTO(CX,N)>3 THEN ERH=1:GOTO 5010
5050 NEXT N
5060 FOR N=CY TO (CY-LGBO*2+2) STEP -2
5070     PTO(CX,N)=CBO
5080     PTO(CX-2,N)=4
5090     PTO(CX+2,N)=4
5100     PTO(CX-2,N-2)=4
5110     PTO(CX+2,N-2)=4
5120 NEXT N
5130 PTO(CX-2,CY+2)=4
5140 PTO(CX,CY+2)=4
5150 PTO(CX+2,CY+2)=4
5160 PTO(CX,CY-LGBO*2)=4
```

```
5170 RETURN
5180 CX=13:CY=61
5190 SOUND 700,2:SOUND 400,3
5200 COLOR 18
5210 LOCATE CX,CY:PRINT CHR$(15)
5220 COLOR 2
5230 A$=INKEY$:IF A$="" THEN 5230
5240 IF A$="8" THEN 5300
5250 IF A$="4" THEN 5390
5260 IF A$="2" THEN 5480
5270 IF A$="6" THEN 5570
5280 IF ASC(A$)=32 THEN 5660
5290 BEEP:GOTO 5230
5300 CX=CX-2
5310 IF CX<3 THEN CX=CX+2:BEEP:GOTO 5370
5320 IF PTO(CX+2,CY)=1 THEN Z=42:GOTO 5360
5330 IF PTO(CX+2,CY)=2 THEN Z=1:GOTO 5360
5340 IF PTO(CX+2,CY)=3 THEN Z=2:GOTO 5360
5350 Z=32
5360 LOCATE CX+2,CY:PRINT CHR$(Z)
5370 LOCATE CX,CY:COLOR 18:PRINT CHR$(15)
5380 GOTO 5220
5390 CY=CY-2
5400 IF CY<53 THEN CY=CY+2:BEEP:GOTO 5460
5410 IF PTO(CX,CY+2)=1 THEN Z=42:GOTO 5450
5420 IF PTO(CX,CY+2)=2 THEN Z=1:GOTO 5450
5430 IF PTO(CX,CY+2)=3 THEN Z=2:GOTO 5450
5440 Z=32
5450 LOCATE CX,CY+2:PRINT CHR$(Z)
5460 LOCATE CX,CY:COLOR 18:PRINT CHR$(15)
5470 GOTO 5220
5480 CX=CX+2
5490 IF CX>21 THEN CX=CX-2:BEEP:GOTO 5550
5500 IF PTO(CX-2,CY)=1 THEN Z=42:GOTO 5540
5510 IF PTO(CX-2,CY)=2 THEN Z=1:GOTO 5540
5520 IF PTO(CX-2,CY)=3 THEN Z=2:GOTO 5540
5530 Z=32
5540 LOCATE CX-2,CY:PRINT CHR$(Z)
5550 LOCATE CX,CY:COLOR 18:PRINT CHR$(15)
5560 GOTO 5220
5570 CY=CY+2
5580 IF CY>71 THEN CY=CY-2:BEEP:GOTO 5640
5590 IF PTO(CX,CY-2)=1 THEN Z=42:GOTO 5630
5600 IF PTO(CX,CY-2)=2 THEN Z=1:GOTO 5630
5610 IF PTO(CX,CY-2)=3 THEN Z=2:GOTO 5630
5620 Z=32
5630 LOCATE CX,CY-2:PRINT CHR$(Z)
5640 LOCATE CX,CY:COLOR 18:PRINT CHR$(15)
5650 GOTO 5220
5660 LOCATE 23,30:PRINT SPACE$(20)
5670 DISPJ=DISPJ+1
5680 LOCATE 15,44:PRINT DISPJ
5690 IF PTO(CX,CY)=0 THEN PTO(CX,CY)=1:LOCATE CX,CY:PRINT CHR$(42):LOCATE 23,39:
PRINT"AGUA":GOTO 5980
5700 IF PTO(CX,CY)=1 THEN LOCATE CX,CY:PRINT CHR$(42):LOCATE 23,34:PRINT"REPETID
O AGUA":CTR=CTR+1:GOTO 5980
5710 IF PTO(CX,CY)=2 THEN LOCATE CX,CY:PRINT CHR$(1):LOCATE 23,32:PRINT"REPETIDO
 TOCADO":CTR=CTR+1:GOTO 5980
5720 IF PTO(CX,CY)=3 THEN LOCATE CX,CY:PRINT CHR$(2):LOCATE 23,32:PRINT"REPETIDO
 HUNDIDO":CTR=CTR+1:GOTO 5980
5730 IF PTO(CX,CY)=4 THEN PTO(CX,CY)=1:LOCATE CX,CY:PRINT CHR$(42):LOCATE 23,39:
PRINT"AGUA":GOTO 5980
5740 PTO(CX,CY)=2
5750 LOCATE CX,CY:PRINT CHR$(1)
5760 FOR I=2 TO 8 STEP 2
5770     IF PTO(CX-I,CY)=4 OR PTO(CX-I,CY)=1 THEN 5810
5780     IF PTO(CX-I,CY)=2 THEN 5800
```

```
5790    GOTO 5970
5800 NEXT I
5810 FOR I=2 TO 8 STEP 2
5820    IF PTO(CX,CY-I)=4 OR PTO(CX,CY-I)=1 THEN 5860
5830    IF PTO(CX,CY-I)=2 THEN 5850
5840    GOTO 5970
5850 NEXT I
5860 FOR I=2 TO 8 STEP 2
5870    IF PTO(CX+I,CY)=4 OR PTO(CX+I,CY)=1 THEN 5910
5880    IF PTO(CX+I,CY)=2 THEN 5900
5890    GOTO 5970
5900 NEXT I
5910 FOR I=2 TO 8 STEP 2
5920    IF PTO(CX,CY+I)=4 OR PTO(CX,CY+I)=1 THEN 5960
5930    IF PTO(CX,CY+I)=2 THEN 5950
5940    GOTO 5970
5950 NEXT I
5960 LOCATE 23,38:PRINT"HUNDIDO":BUNO=BUNO+1:RPTIRJ=1:GOTO 6000
5970 LOCATE 23,38:PRINT"TOCADO":RPTIRJ=1
5980 LOCATE 17,45:PRINT BUNO
5990 RETURN
6000 LOCATE CX,CY:PRINT CHR$(2):PTO(CX,CY)=3
6010 FOR I=2 TO 8 STEP 2
6020    IF PTO(CX-I,CY)=2 THEN 6040
6030    GOTO 6060
6040    PTO (CX-I,CY)=3:LOCATE CX-I,CY:PRINT CHR$(2)
6050 NEXT I
6060 FOR I=2 TO 8 STEP 2
6070    IF PTO(CX,CY-I)=2 THEN 6090
6080    GOTO 6110
6090    PTO(CX,CY-I)=3:LOCATE CX,CY-I:PRINT CHR$(2)
6100 NEXT I
6110 FOR I=2 TO 8 STEP 2
6120    IF PTO(CX+I,CY)=2 THEN 6140
6130    GOTO 6160
6140    PTO(CX+I,CY)=3:LOCATE CX+I,CY:PRINT CHR$(2)
6150 NEXT I
6160 FOR I=2 TO 8 STEP 2
6170    IF PTO(CX,CY+I)=2 THEN 6190
6180    GOTO 6210
6190    PTO(CX,CY+I)=3:LOCATE CX,CY+I:PRINT CHR$(2)
6200 NEXT I
6210 GOTO 5980
6220 DISPO=DISPO+1:LOCATE 3,45:PRINT DISPO
6230 LOCATE 10,38:PRINT"A/T/H":LOCATE 7,44:PRINT"     "
6240 IF  BTOC=1 THEN 7430
6250 IF DISPO>=45 THEN 7780
6260 CXO=INT(RND*18)+3
6270 IF CXO MOD 2 =0 THEN 6260
6280 CYO=INT(RND*18)+9
6290 IF CYO MOD 2 =0 THEN 6280
6300 IF PTO(CXO,CYO)>0 AND PTO(CXO,CYO)<4 THEN 6260
6310 IF (PTO(CXO,CYO)=0)OR(PTO(CXO,CYO)=4) THEN PTO(CXO,CYO)=1:PTOAG=1:GOTO 6620
6320 PTO(CXO,CYO)=2
6330 PTO(CXO-2,CYO-2)=1
6340 PTO(CXO+2,CYO-2)=1
6350 PTO(CXO-2,CYO+2)=1
6360 PTO(CXO+2,CYO+2)=1
6370 FOR I=2 TO 8 STEP 2
6380    IF PTO(CXO-I,CYO)=4 OR PTO(CXO-I,CYO)=1 THEN 6420
6390    IF PTO(CXO-I,CYO)=2 THEN 6410
6400    GOTO 6600
6410 NEXT I
6420 FOR I=2 TO 8 STEP 2
6430    IF PTO(CXO,CYO-I)=4 OR PTO(CXO,CYO-I)=1 THEN 6470
6440    IF PTO(CXO,CYO-I)=2 THEN 6460
6450    GOTO 6600
```

```
6460 NEXT I
6470 FOR I=2 TO 8 STEP 2
6480    IF PTO(CXO+I,CYO)=4 OR PTO(CXO+I,CYO)=1 THEN 6520
6490    IF PTO(CXO+I,CYO)=2 THEN 6510
6500    GOTO 6600
6510 NEXT I
6520 FOR I=2 TO 8 STEP 2
6530    IF PTO(CXO,CYO+I)=4 OR PTO(CXO,CYO+I)=1 THEN 6570
6540    IF PTO(CXO,CYO+I)=2 THEN 6560
6550    GOTO 6600
6560 NEXT I
6570 BUNJ=BUNJ+1:BTOC=0:BHUN=1:PTO(CXO,CYO)=3
6580 SOUND 600,3:SOUND 400,3:SOUND 500,5:SOUND 600,4:SOUND 400,6
6590 GOTO 6620
6600 BTOC=1:RPTIRO=1:CXTC=CXO:CYTC=CYO:PTOTOC=1
6610 REM
6620 REM *** PREGUNTA EL RESULTADO DEL DISPARO ***
6630 REM
6640 SOUND 100,3:SOUND 500,2
6650 CLT=64
6660 FOR FIL=3 TO CXO STEP 2
6670    CLT=CLT+1
6680 NEXT FIL
6690 LOCATE 7,44:PRINT CHR$(CLT)
6700 CNM=48
6710 IF CYO=27 THEN 6750
6720 FOR COL=9 TO CYO STEP 2
6730    CNM=CNM+1
6740 NEXT COL
6750 LOCATE 7,46:PRINT CHR$(CNM)
6760 COLOR 18
6770 LOCATE 9,44:PRINT CHR$(63)
6780 A$=INKEY$:IF A$="" THEN 6780
6790 LOCATE 9,44:PRINT" "
6800 IF A$="A" OR A$="a" THEN LOCATE 10,38:PRINT"A":COLOR 2:GOTO 6840
6810 IF A$="T" OR A$="t" THEN LOCATE 10,40:PRINT"T":COLOR 2:GOTO 6860
6820 IF A$="H" OR A$="h" THEN LOCATE 10,42:PRINT"H":COLOR 2:GOTO 6880
6830 BEEP:GOTO 6780
6840 IF PTOAG=1 THEN PTOAG=0:Z=42:GOTO 6900
6850 GOTO 6920
6860 IF PTOTOC=1 THEN PTOTOC=0:Z=1:GOTO 6900
6870 GOTO 6920
6880 IF BHUN=1 THEN BHUN=0:GOTO 7090
6890 GOTO 6920
6900 LOCATE CXO,CYO:PRINT CHR$(Z)
6910 RETURN
6920 LOCATE 11,30:PRINT"(ESTAS SEGURO?(S/N)"
6930 A$=INKEY$:IF A$="" THEN 6930
6940 IF A$="S" OR A$="s" THEN 7060
6950 IF A$="N" OR A$="n" THEN 6970
6960 BEEP:GOTO 6930
6970 LOCATE 9,29:PRINT"No trates de enga$arme"
6980 LOCATE 11,30:PRINT SPACE$(20)
6990 LOCATE 10,30:PRINT"Pulsa una tecla"
7000 A$=INKEY$:IF A$="" THEN 7000
7010 LOCATE 9,29:PRINT SPACE$(22)
7020 LOCATE 10,30:PRINT SPACE$(20)
7030 LOCATE 9,34:PRINT"RESULTADO"
7040 LOCATE 10,38:PRINT"A/T/H"
7050 GOTO 6620
7060 CLS
7070 LOCATE 10,30:PRINT"NO JUEGO CON TRAMPOSOS"
7080 GOTO 1740
7090 LOCATE CXO,CYO:PRINT CHR$(2)
7100 FOR I=2 TO 8 STEP 2
7110    IF PTO(CXO-I,CYO)=2 THEN 7130
```

```
7120    GOTO 7160
7130    PTO(CXO-I,CYO)=3:LOCATE CXO-I,CYO:PRINT CHR$(2)
7140    IF PTO(CXO-I-2,CYO)=4 THEN PTO(CXO-I-2,CYO)=1
7150 NEXT I
7160 FOR I=2 TO 8 STEP 2
7170    IF PTO(CXO,CYO-I)=2 THEN 7190
7180    GOTO 7220
7190    PTO(CXO,CYO-I)=3:LOCATE CXO,CYO-I:PRINT CHR$(2)
7200    IF PTO(CXO,CYO-I-2)=4 THEN PTO(CXO,CYO-I-2)=1
7210 NEXT I
7220 FOR I=2 TO 8 STEP 2
7230    IF PTO(CXO+I,CYO)=2 THEN 7250
7240    GOTO 7280
7250    PTO(CXO+I,CYO)=3:LOCATE CXO+I,CYO:PRINT CHR$(2)
7260    IF PTO(CXO+I+2,CYO)=4 THEN PTO(CXO+I+2,CYO)=1
7270 NEXT I
7280 FOR I=2 TO 8 STEP 2
7290    IF PTO(CXO,CYO+I)=2 THEN 7310
7300    GOTO 7340
7310    PTO(CXO,CYO+I)=3:LOCATE CXO,CYO+I:PRINT CHR$(2)
7320    IF PTO(CXO,CYO+I+2)=4 THEN PTO(CXO,CYO+I+2)=1
7330 NEXT I
7340 RPTIRO=1
7350 PTO(CXO-2,CYO)=1
7360 PTO(CXO+2,CYO)=1
7370 PTO(CXO,CYO-2)=1
7380 PTO(CXO,CYO+2)=1
7390 LOCATE 5,45:PRINT BUNJ
7400 SOUND 800,2:SOUND 1000,2:SOUND 700,2:SOUND 100,3
7410 GOTO 6910
7420 REM
7430 REM *** INVESTIGA EL ORDENADOR PARA VOLVER A DISPARAR ***
7440 REM
7450 DBQ=INT(RND*3)+1
7460 ON DBQ GOTO 7470,7490,7510,7530
7470 IF ((CXTC-2)<3) OR ((PTO(CXTC-2,CYTC)>0) AND (PTO(CXTC-2,CYTC)<4)) THEN 749
0
7480 CXO=CXTC-2:CYO=CYTC:GOTO 6310
7490 IF ((CYTC-2)<9) OR ((PTO(CXTC,CYTC-2)>0) AND (PTO(CXTC,CYTC-2)<4)) THEN 751
0
7500 CXO=CXTC:CYO=CYTC-2:GOTO 6310
7510 IF ((CXTC+2)>21) OR ((PTO(CXTC+2,CYTC)>0) AND (PTO(CXTC+2,CYTC)<4)) THEN 75
30
7520 CXO=CXTC+2:CYO=CYTC:GOTO 6310
7530 IF ((CYTC+2)>27) OR ((PTO(CXTC,CYTC+2)>0) AND (PTO(CXTC,CYTC+2)<4)) THEN 75
50
7540 CXO=CXTC:CYO=CYTC+2:GOTO 6310
7550 SOUND 100,2:SOUND 220,2
7560 FOR I=2 TO 4 STEP 2
7570    IF (CXTC-I)<3 THEN 7600
7580    IF PTO(CXTC-I,CYTC)>4 THEN CXO=CXTC-I:GOTO 7730
7590 NEXT I
7600 FOR I=2 TO 4 STEP 2
7610    IF (CXTC+I)>21 THEN 7640
7620    IF PTO(CXTC+I,CYTC)>4 THEN CXO=CXTC+I:GOTO 7730
7630 NEXT I
7640 FOR I=2 TO 4 STEP 2
7650    IF (CYTC-I)<9 THEN 7680
7660    IF PTO(CXTC,CYTC-I)>4 THEN CYO=CYTC-I:GOTO 7750
7670 NEXT I
7680 FOR I=2 TO 4 STEP 2
7690    IF (CYTC+I)>27 THEN 7730
7700    IF PTO(CXTC,CYTC+I)>4 THEN CYO=CYTC+I:GOTO 7750
7710 NEXT I
7720 GOTO 6260
7730 CYO=CYTC
```

```
7740 GOTO 6310
7750 CXO=CXTC
7760 GOTO 6310
7770 REM
7780 REM *** BARRE LA PANTALLA EN BUSCA DE DISPAROS ***
7790 REM
7800 FOR BUSC1= 3 TO 21 STEP 2
7810    FOR BUSC2= 9 TO 27 STEP 2
7820       IF (PTO(BUSC1,BUSC2)=0) OR (PTO(BUSC1,BUSC2)>3) THEN 7870
7830    NEXT BUSC2
7840 NEXT BUSC1
7850 SOUND 2000,1:SOUND 1000,1:SOUND 3000,1
7860 GOTO 7780
7870 CXO=BUSC1:CYO=BUSC2
7880 GOTO 6310
```

El programa que proponemos a continuación se realizó bajo GWBASIC, en un IBM.

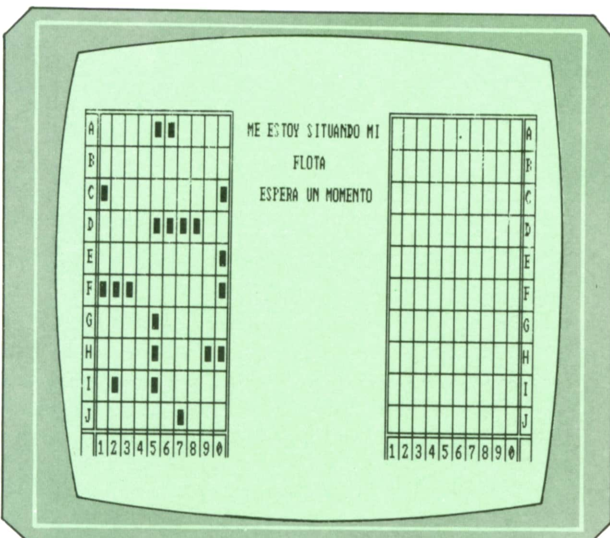
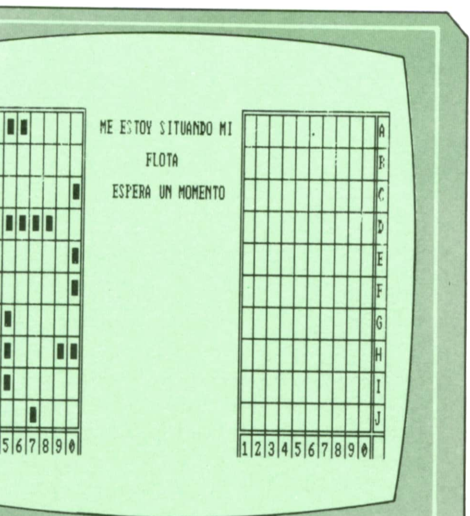




Fig. 2. El ordenador coloca sus gráficos.

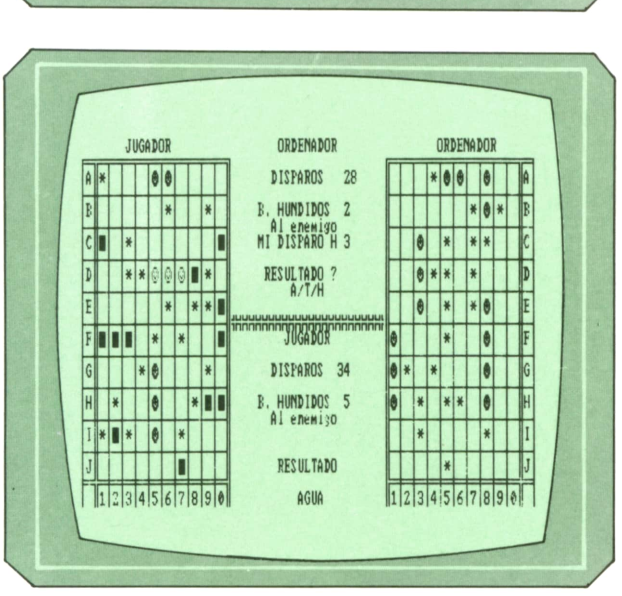



Fig. 3. Jugando.



Programa: Frontón para SPECTRUM

Con el programa que mostramos a continuación queremos realizar una sencilla versión del famoso BREAKOUT. Este juego fue uno de los primeros que aparecieron en las máquinas de juegos electrónicos de los billares.

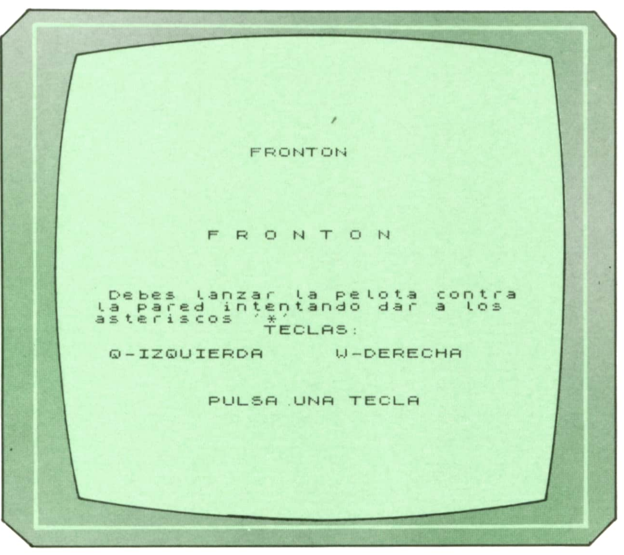



Fig. 4.

El juego consiste en destruir la muralla que nos cierra el camino de la libertad y que nos separa del mundo exterior. Para ello contamos con una bola que va rebotando por las paredes de la pantalla y que, cada vez que toca un ladrillo, lo pulveriza.

Para cambiar la dirección de la bola utilizaremos una raqueta, que aparecerá en la parte inferior de la pantalla y que moveremos con las siguientes teclas:

Q - Izquierda
W - Derecha

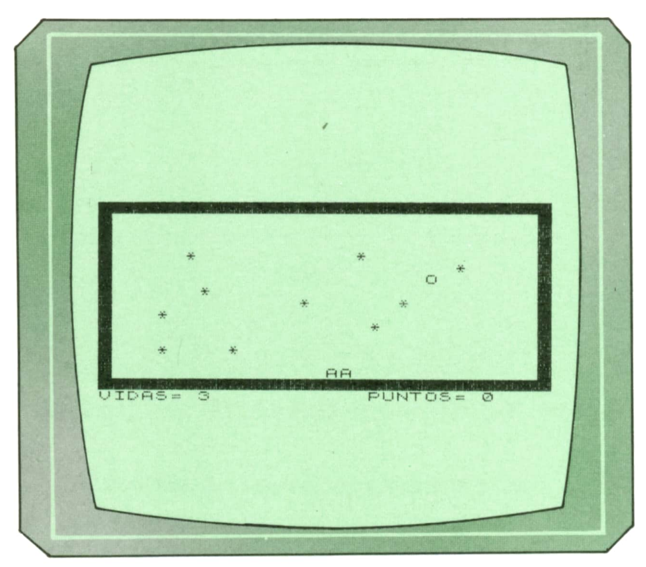




Fig. 5.

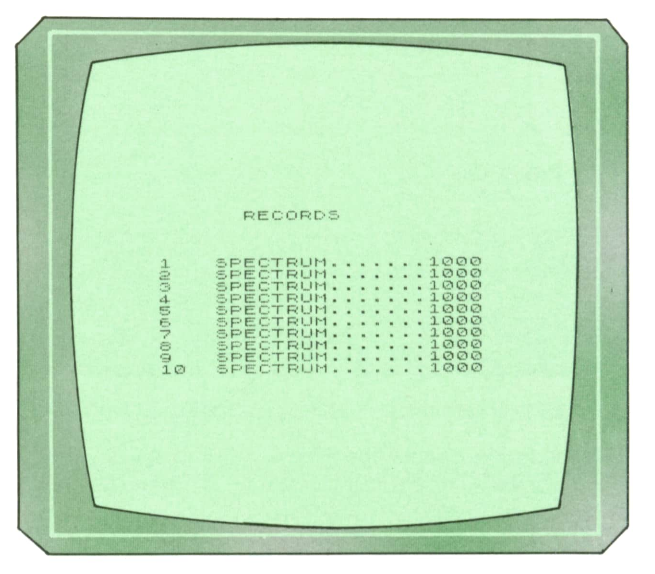
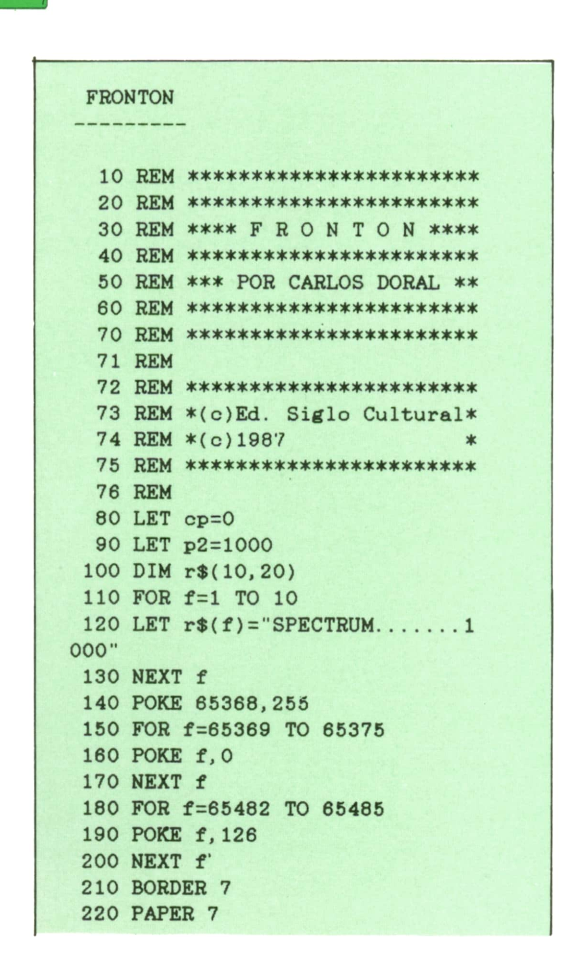




Fig. 6. Tabla de récords.



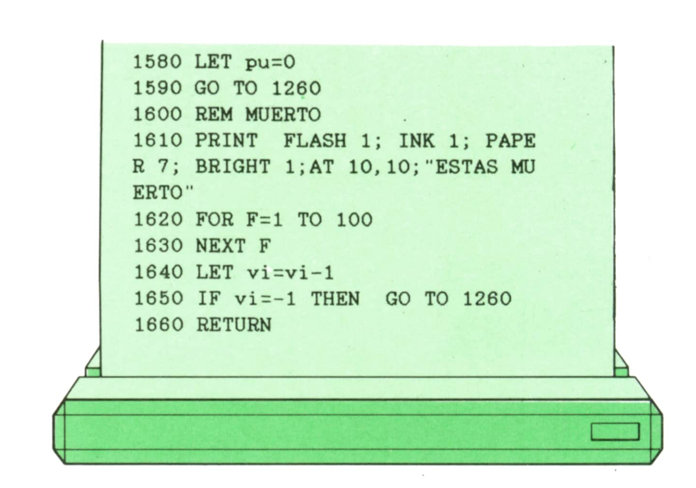
```
 FRONTON
---------

  10 REM ***********************
  20 REM ***********************
  30 REM **** F R O N T O N ****
  40 REM ***********************
  50 REM *** POR CARLOS DORAL **
  60 REM ***********************
  70 REM ***********************
  71 REM
  72 REM ***********************
  73 REM *(c)Ed. Siglo Cultural*
  74 REM *(c)1987             *
  75 REM ***********************
  76 REM
  80 LET cp=0
  90 LET p2=1000
 100 DIM r$(10,20)
 110 FOR f=1 TO 10
 120 LET r$(f)="SPECTRUM.......1
000"
 130 NEXT f
 140 POKE 65368,255
 150 FOR f=65369 TO 65375
 160 POKE f,0
 170 NEXT f
 180 FOR f=65482 TO 65485
 190 POKE f,126
 200 NEXT f
 210 BORDER 7
 220 PAPER 7
```

```
 230 INK 0
 240 CLS
 250 PRINT  INK 1; FLASH 1; BRIG
HT 1;AT 0,11;" FRONTON "
 260 PRINT AT 12,0;" Debes lanz
ar la pelota contra  la pared in
tentando dar a los   asteriscos
'*'
 TECLAS:
                       Q-IZQUIERD
A     W-DERECHA"
 270 PRINT  FLASH 1;AT 21,9;"PUL
SA UNA TECLA"
 280 LET a$=" F R O N T O N "
 290 PRINT  FLASH 1; INK 2; BRIG
HT 1; INK 2;AT 7,8;a$
 300 LET co=0
 310 LET l=1
 320 LET y1=7
 330 LET y2=10
 340 LET a=1
 350 FOR r=1 TO 2
 360 LET a$=" F R O N T O N "
 370 FOR c=1 TO LEN a$
 380 FOR f=y1 TO y2 STEP a
 390 LET b$=a$(l TO l)
 400 PRINT AT f,l+7; FLASH 1; IN
K 2; BRIGHT 1;b$;AT f-a,l+7; FLA
SH 0; BRIGHT 0;" "
 410 IF INKEY$<>"" THEN  GO TO 5
30
 420 LET co=co+1
 430 IF co>119 AND cp=1 THEN  LE
```

```
T co=0: GO TO 1260
 440 NEXT f
 450 LET l=l+1
 460 NEXT c
 470 LET y1=10
 480 LET y2=7
 490 LET a=-1
 500 LET l=1
 510 NEXT r
 520 GO TO 300
 530 LET cp=1
 540 LET di=1
 550 LET ma=di*10
 560 LET pu=0
 570 LET vi=3
 580 CLS
 590 FOR I=0 TO 31: PRINT CHR$ 1
43;: NEXT I
 600 PRINT AT 15,0;: FOR I=0 TO
31: PRINT CHR$ 143;: NEXT I
 610 FOR f=1 TO 14
 620 PRINT AT f,0;CHR$ 143;AT f,
31;CHR$ 143
 630 NEXT f
 640 FOR f=18 TO 21
 650 PRINT AT f,0; PAPER f-16;"
                               "
 660 NEXT f
 670 PRINT AT 16,0;"VIDAS= ";vi
 680 PRINT AT 16,19;"PUNTOS= ";p
u
 690 LET ch=0
 700 LET x=15
 710 LET y=14
 720 LET a=2
 730 LET b=4+INT (RND*22)
 740 LET xx=1
 750 LET yy=1
 760 FOR f=1 TO ma
 770 PRINT  INK 1;AT 3+INT (RND*
10),3+INT (RND*25);"*"
 780 NEXT f
 790 PRINT  INK 3;AT a,b;" "; IN
K 0;AT a-yy,b-xx;" "
 800 FOR f=1 TO 100
 810 NEXT f
 820 PRINT AT y,x;" ";CHR$ 144;C
HR$ 144;" "
 830 POKE 23658,8
 840 PRINT AT 16,27;pu
 850 IF INKEY$="Q" AND x>2 THEN
 LET x=x-1
 860 LET ch=ch+1
 870 IF ch>99 AND a<5 AND b<26 A
ND SGN xx=1 AND b>4 THEN  PRINT
AT a+1,b-1;" ": LET b=b+1: LET c
h=0
 880 IF INKEY$="W" AND x<27 THEN
   LET x=x+1
 890 IF ATTR (a,b)=57 THEN  LET
pu=pu+5: LET ma=ma-1: LET ch=0:
IF ma=0 THEN  GO TO 990
 900 PRINT  INK 3;AT a,b;" "; IN
K 0;AT a-yy,b-xx;" "
 910 LET a=a+yy
 920 LET b=b+xx
 930 IF ATTR (y-1,x+1)=59 OR ATT
R (y-1,x+2)=59 THEN  LET pu=pu+1
: BEEP .05,-30: LET yy=-yy: PRIN
T AT y-1,x-1;"    ": LET a=a-2
 940 IF b>28 THEN  LET pu=pu+1:
LET xx=-xx: PRINT AT a-yy,b+xx;"
 ": BEEP .05,10
 950 IF b<3 THEN  LET pu=pu+1: L
ET xx=-xx: PRINT AT a-yy,b+xx;"
": BEEP .05,10
 960 IF a>14 THEN  GO SUB 1600:
GO TO 580
 970 IF a<3 THEN  LET pu=pu+1: B
EEP .05,0: LET yy=-yy: PRINT AT
a+yy,b-xx;" "
 980 GO TO 820
 990 REM PASO DE NIVEL
1000 CLS
1010 LET di=di+1
1020 PRINT  FLASH 1;AT 10,5;"PAS
AS A SIGUIENTE NIVEL"
1030 PRINT AT 13,10;"PUNTOS= ";p
u
1040 FOR f=1 TO 400
1050 NEXT f
1060 FOR f=pu TO pu+500 STEP 10
1070 PRINT AT 13,18;f
1080 BEEP .01,0
1090 NEXT f
1100 LET pu=pu+500
1110 FOR f=1 TO 300
1120 NEXT f
1130 LET ma=di*10
1140 LET vi=vi+1
1150 GO TO 580
1160 REM RECORDS
1170 LET co=0
1180 CLS
1190 PRINT  FLASH 1;AT 0,12;"REC
ORDS"
1200 FOR f=1 TO 10
1210 PRINT  INK INT (RND*6);AT f
+3,6;f;AT f+3,10;r$(f)
1220 NEXT f
1230 LET co=co+1
1240 IF co=20 THEN  GO TO 210
1250 GO TO 1200
1260 REM FINAL
1270 CLS
1280 PRINT  FLASH 1;AT 0,12;"REC
ORDS"
1290 FOR f=1 TO 10
1300 PRINT  INK INT (RND*6);AT f
+3,6;f;AT f+3,10;r$(f)
1310 NEXT f
1320 LET le=LEN STR$ pu
1330 FOR f=1 TO 10
1340 LET a$=r$(f)
1350 IF pu>=p2 THEN  GO TO 1400
1360 NEXT f
1370 FOR x=1 TO 300
1380 NEXT x
1390 GO TO 210
1400 REM INPUT
1410 INPUT "Nombre: "; LINE n$
1420 IF LEN n$<1 OR LEN n$>13 TH
EN  GO TO 1410
1430 LET l=LEN n$+le
```

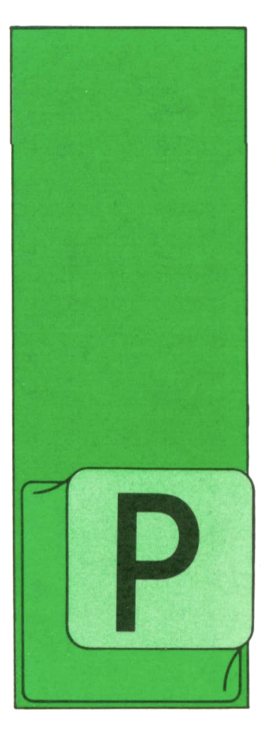
```
1440 LET l=20-l
1450 LET b$=""
1460 FOR x=1 TO (l-1)
1470 LET b$=b$+"."
1480 NEXT x
1490 LET r$(10)=n$+b$+STR$ pu
1500 LET p2=pu
1510 FOR c=1 TO 9
1520 FOR f=9 TO 1 STEP -1
1530 LET c$=r$(f)
1540 LET e$=r$(f+1)
1550 IF c$(20-le TO )<e$(20-le T
O ) THEN  LET c$=r$(f): LET r$(f
)=r$(f+1): LET r$(f+1)=c$
1560 NEXT f
1570 NEXT c
```



```
1580 LET pu=0
1590 GO TO 1260
1600 REM MUERTO
1610 PRINT  FLASH 1; INK 1; PAPE
R 7; BRIGHT 1;AT 10,10;"ESTAS MU
ERTO"
1620 FOR F=1 TO 100
1630 NEXT F
1640 LET vi=vi-1
1650 IF vi=-1 THEN  GO TO 1260
1660 RETURN
```

TECNICAS DE ANALISIS

BASES DE DATOS CON TABLAS INVERTIDAS DE INDICES Y EN RED



Bases de datos invertidas

PARA resolver algunos de los problemas que aparecen en las bases de datos jerárquicas se puede utilizar un SGBD con el modelo de tablas de índices invertidas, en el que los datos aparecen sin jerarquizar y se añaden unas tablas de índices que faciliten las consultas y las manipulaciones de los datos.

Consideramos la siguiente información bibliográfica:

1WIRTH, N. Alg. + Estruc. datos = Programas 382 Casti 1980 1200 Díaz Sant.

2WIRTH, N. The design of a Pascal Compiler 150 Amme. 1971 98FF Lib. Uni.

3HARTNELL Int. Art.: conceptos y programas 267 Anaya 1985 1500 Lib. Tecni.

4HARTNELL Libro Gig. Jueg. para ZX Spectrum 310 Anaya 1986 1750 Edic. Perg.

5HARTNELL Libro Gig. Juego. para ordenador 259 Anaya 1984 1500 Lib. Tecni.

6GRIES, D. Compiler Cnst. for Dig. Computers 452 JhonW 1984 32$$ Thec. Lib.

Se pueden construir tablas invertidas de índices para los diferentes conceptos en que están divididos los registros de más arriba.

Así, por ejemplo, podemos construir una tabla de índices para la fecha de edición, otra para la editorial, etc. Quedarían del siguiente modo las dos indicadas:

INDICE	FECHA
1971	2
1980	1
1984	5,6
1985	3
1986	4

INDICE	EDITORIAL
Amms.	2
Anaya	3,4,5
Casti	1
JohnW	6

Por este procedimiento, si queremos averiguar cuántos libros (y cuáles) de nuestra biblioteca fueron publicados en 1984, mediante una sencilla inspección de la tabla del concepto FECHA vemos que para el índice 1984 hay dos «entradas» (anotaciones): la 5 y la 6, correspondientes a un libro de Hartnell y otro de Gries.

Incluso la pregunta «¿cuántos libros editados antes de 1984 tenemos?», puede contestarse mucho más fácilmente manejando los datos de la tabla de índices de FECHA, que mediante examen de toda la base de datos.

Pueden, incluso, manejarse dos tablas de índices para procesarse una pregunta compuesta (del tipo «¿cuántos volúmenes tenemos de la editorial Anaya publicados antes de 1984?».

Naturalmente, sólo se pueden hacer este tipo de consultas rápidas y eficaces si previamente se ha creado la tabla de índices adecuada: si la estructura de la base es estable y el tipo de consultas previsible, esto no es ningún problema; pero sí lo es si la estructura de los datos es modificable y/o las consultas y manipulaciones imprevisibles.

Por otro lado, la creación de índices (de muchos índices) supone una redundancia de la información indeseable en numerosas ocasiones. Además, las tablas de índices suelen ser heterogéneas, (índices con muchas entradas y otros con pocas), lo que hace que las búsquedas no sean uniformes en tiempos y que, en ocasiones, dependa mucho la demora en la respuesta del orden en que se procesen los datos (según las longitudes de las listas que se examinen antes).

Normalmente, la base de datos con tablas de índices invertidas suele estar organizada en tres grandes zonas:

— *Indices.* Relación de los diferentes conceptos de búsqueda y referencia de las tablas de valores de dichos conceptos.

INDICES PRIMARIOS

TABLA DE INDICES

TABLA DE DIRECCIONES

CONSULTA

INDICE VALOR PUNTERO LONGITUD

VALOR 1 DIRECCIONES

VALOR 2 DIRECCIONES

BASE DE DATOS

INDICE VALOR PUNTERO LONGITUD

(OPCIONAL)



Organización de una base de datos con tablas invertidas de índices.

— *Tablas de direcciones.* Donde se reseñan las direcciones de los registros que tienen cada valor para los diferentes conceptos.

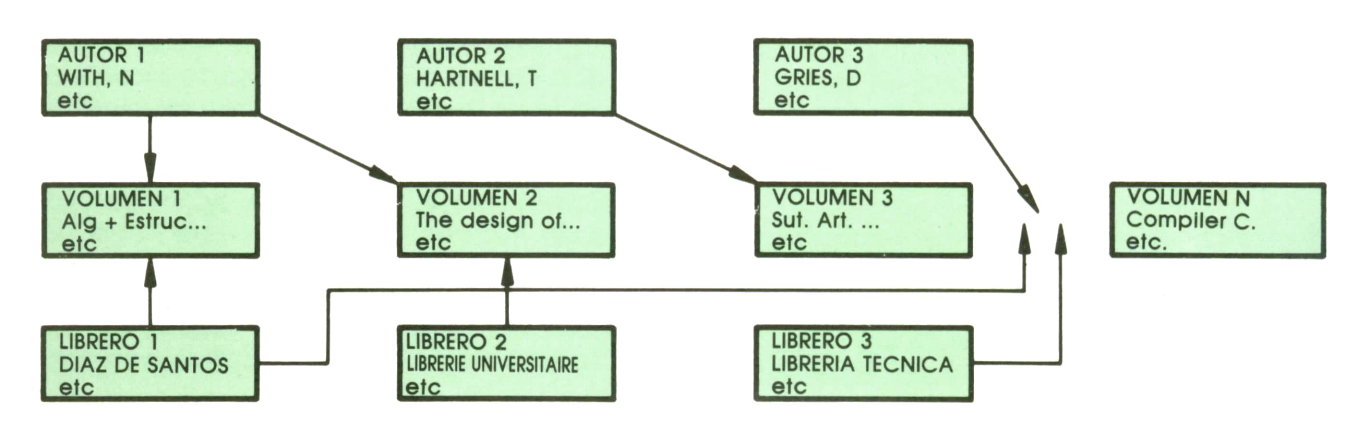
— *Base de datos* propiamente dicha. En la que se encuentran las informaciones almacenadas.

En ocasiones se suelen estructurar las tablas de índices en diversas subtablas, para realizar las búsquedas en varias etapas (a varios niveles): primero se examina la tabla de los conceptos de búsqueda para localizar la tabla de valores del concepto que interese y, posteriormente, se inspecciona la tabla de valores para localizar el buscado.

Bases de datos con estructura de red

En las bases de datos en red, los diferentes grupos de informaciones se almacenan separadamente, sin referencia alguna a las relaciones entre ellas.

Separadamente se mantienen tablas que relacionan unos conceptos con otros; cada elemento puede tener varios superiores (el elemento superior en una relación se llama *propietario-owner* y el elemento inferior *miembro-member);* cada elemento puede tener varios miembros.



Las relaciones se suelen mantener mediante «punteros» o «indicadores» referenciables como parejas de números: la pareja (m,n) representa que m es owner en una relación de la que es member n.

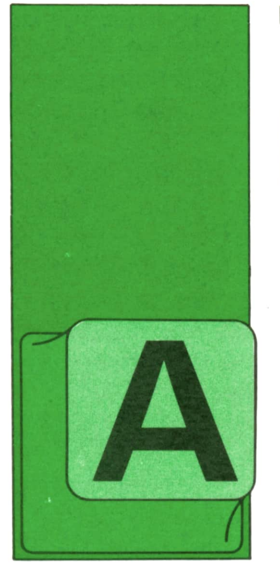
Así, en el caso anterior podemos establecer un conjunto de registros de *autores,* otros registros de *volúmenes* y un grupo de registros de *libreros:* habrá que establecer, por otro lado, las relaciones pertinentes entre cada autor (o autores) y el volumen (o volúmenes) de los que son titulares; también habrá que relacionar cada librero con todos los volúmenes que hemos adquirido en su establecimiento... y del mismo modo relacionar cada editorial con los libros que ha editado, etc.

La estructura resultante de este modelo es mucho más compleja y las búsquedas, en ocasiones, pueden no ser tan directas, pero el conjunto es sumamente flexible y libera al usuario de tener que considerar, cuando maneja los datos, las relaciones existentes entre los diferentes conjuntos de informaciones.

TECNICAS DE PROGRAMACION



Manejo de ficheros



A menudo es preciso construir programas que no son completos en sí mismos, sino que para ejecutarse correctamente deben obtener información de diversas fuentes, la más frecuente de las cuales es un fichero en disco fijo o flexible (disquete). También es posible que un programa genere cierta información que debe guardarse, sin desaparecer al final del programa (como normalmente ocurre), y es frecuente que dicha información se guarde también en un fichero en disco.

Así, pues, un fichero en disco es un conjunto de información que queremos conservar permanentemente, con independencia de la ejecución del programa. Además, los ficheros son también útiles por otros motivos, de los que vamos a enumerar los siguientes:

— Cuando los mismos datos deben ser utilizados por varios programas diferentes.

— Cuando los datos son tan abundantes que no cabrían en la memoria principal junto con el programa, pero sí se pueden fraccionar para su tratamiento en unidades más pequeñas, que llamaremos registros.

Veamos algunos casos de programas de aplicación en los que es imprescindible trabajar con ficheros:

— Editores de textos.

— Procesadores de textos.

— Bases de datos.

— Programas de contabilidad.

— Gestión de personal de una empresa.

— Hojas de cálculo.



Operaciones con ficheros

Supongamos que hemos decidido utilizar uno o varios ficheros en nuestro programa. Podremos realizar con él las siguientes operaciones:

1. Crear el fichero.
2. Prepararlo para su utilización una vez creado ("abrir" el fichero).
3. Leer los datos contenidos en el fichero.
4. Escribir datos nuevos en el fichero.
5. Decirle al programa que ya no necesitamos más este fichero ("cerrar" el fichero).



Operaciones con ficheros en BASIC

En el lenguaje BASIC, las instrucciones utilizadas para realizar operaciones con ficheros no siempre son las mismas, sino que a menudo dependen del intérprete con el que estemos trabajando. En las líneas sucesivas vamos a hacer referencia a las instrucciones válidas para ordenadores IBM PC o compatibles.

Las dos primeras operaciones (creación de un fichero y apertura de uno antiguo) se realizan con la instrucción siguiente:

```
OPEN "nombre" AS #número
```

donde la palabra inglesa OPEN significa «abrir», donde "nombre" es el nombre del fichero (debe escribirse entre comillas dobles) y número es el número de orden que asignamos a este fichero, entre los varios que pueden abrirse simultáneamente desde nuestro programa. Si sólo tenemos un fichero abierto (como es frecuente), bastará con que utilicemos el número 1.

El nombre que se le puede dar a un fichero depende del sistema operativo de que dispongamos. Si se trata del DOS (probablemente el más utilizado), el nombre de un fichero consta de tres partes:

```
dispositivo:nombre.extensión
```

«Dispositivo» es una letra que indica al sistema operativo en qué unidades de disco o disquete está el fichero. Normalmente será A, B o C. Después de dicha letra deben venir dos puntos. Pero si no se indica la letra del dispositivo (y en ese caso tampoco escribiremos los dos puntos), se entiende que el fichero tiene que estar en la unidad que el sistema utiliza por defecto.

«Nombre» es un nombre cualquiera, de una a ocho letras o cifras, excepto uno de los siguientes: LPT1, LPT2, CON, COM, AUX.

«Extensión» es un nombre cualquiera, de una a tres letras o cifras, que puede faltar. Si no damos la extensión, tampoco tendremos que escribir el punto de separación.

Veamos algunos ejemplos de nombres válidos para ficheros en disco o disquete:

C:27UUU.DDD
B:Q
FICHERO1.BB
A

En cambio, los siguientes nombres no son válidos:

C.27UUU:DDD (el punto y los dos puntos están mal colocados)
U:Q (no existe la unidad U)
B:COM.BB (COM es uno de los nombres reservados del DOS)
ABCDEFGHI.A (más de 8 caracteres en el nombre)

Al abrir un fichero con la instrucción OPEN, podemos elegir para qué queremos abrirlo. Puede ser para realizar una de las operaciones siguientes:

— Para leer secuencialmente los registros del fichero. En este caso la instrucción que abre el fichero debe escribirse así:

OPEN "nombre" FOR INPUT AS #número

— Para escribir secuencialmente registros en el fichero. En este caso tendremos que usar la siguiente variante de la instrucción OPEN:

OPEN "nombre" FOR OUTPUT AS # número

— Para añadir registros nuevos al final de un fichero preexistente. Diremos entonces:

OPEN "nombre" FOR APPEND AS # número

— Para leer o escribir registros en cualquier posición del fichero (lectura y escritura directa). En este caso puede utilizarse la instrucción OPEN en su forma más sencilla:

OPEN "nombre" AS #número

Este último caso es más complicado y no lo vamos a detallar aquí.

Al ejecutarse una instrucción OPEN pueden ocurrir las siguientes circunstancias:

1. Que se abra el fichero para lectura secuencial. En tal caso, si el fichero no existía, obtendremos un mensaje de error. Si existía, la operación se llevará a cabo correctamente y el fichero quedará preparado para la lectura de los datos.

2. Que se abra el fichero para escritura secuencial, para añadir registros o para lectura y escritura directa. En este caso, si el fichero no existe, será creado automáticamente. En cualquier caso, quedará preparado para realizar las operaciones correspondientes.

El procedimiento para leer datos de un fichero es muy semejante al que se utiliza para leerlos del teclado (con la instrucción INPUT), que ya hemos explicado en capítulos anteriores. De hecho, son válidas todas las formas de la instrucción INPUT que vimos allí, con la única salvedad de que hay que especificar el número del fichero del que queremos leer los datos (el mismo número que se dio en la instrucción OPEN). Por tanto, la forma más general de la instrucción INPUT es:

INPUT #número,variable,variable,...

que lee del fichero que tiene el número indicado los valores que habrá que asignar a las variables especificadas en el resto de la instrucción (cuyos nombres están separados por comas).

De igual manera, para escribir datos en un fichero puede utilizarse la instrucción PRINT en todas sus formas, especificando también en este caso el número del fichero en el que queremos escribir. De esta manera, la forma más general de la instrucción PRINT será:

PRINT #número,expresión;expresión;...

que escribe en el fichero que tiene el número indicando los valores de las expresiones especificadas en el resto de la instrucción (separadas por puntos y comas).

Finalmente, para decirle al programa que ya no necesitamos más cierto fichero (para "cerrar" el fichero) utilizaremos la siguiente instrucción:

CLOSE #número

Vamos a ver un ejemplo.

Supongamos que queremos pasar cierto número de datos de un programa a otro diferente. Esto puede ocurrir, por ejemplo, si el primer programa calcula cierta información que el segundo utilizará más tarde para realizar nuevos cálculos u operaciones. En el número 18 de *Informática y Programación Paso a Paso,* en la sección de Programas, aparece un programa diseñador de juegos de aventura que consta de dos partes distintas que se procesan independientemente. La primera permite crear un juego de aventura; la segunda ejecutarlo. Pues bien: los datos que permiten ejecutar una aventura determinada son colocados por el primer programa en un fichero intermedio, que el segundo programa leerá y utilizará para obtener los valores de ciertas variables.

Veamos un ejemplo muy sencillo: supongamos que tenemos un programa que debe pasarle datos a otro programa diferente. Estos datos serán los valores de ciertas variables del primer programa, llamadas I, X$ y W, donde I es una variable numérica escalar, X$ es una cadena de caracteres y W es una tabla o matriz de 3 filas y 5 columnas. El programa siguiente crea un fichero y escribe en él los valores de dichas variables:

Obsérvese lo siguiente:

1. El fichero donde vamos a guardar las variables se llamará «FICHERO». En la instrucción 9020 lo abrimos para escribir (OUTPUT) y le asignamos el número 1.

```
10 DIM W(3,5)
. . . . . . .
9000 REM Guardar los valores de ciertas
9010 REM variables en un fichero
9020 OPEN "FICHERO" FOR OUTPUT AS #1
9030 PRINT #1,I;X$
9040 FOR J=1 TO 3
9050 FOR K=1 TO 5
9060 PRINT #1,W(J,K);
9070 NEXT K
9080 PRINT #1,""
9090 NEXT J
9100 CLOSE #1
. . . . . . .
```

2. En una sola instrucción (la 9030) imprimimos las variables I y X$, sin separarlas por un paso de línea (las dos variables ocuparán conjuntamente un registro del fichero).

3. Para escribir la variable W necesitamos un doble bucle: uno para las filas y otro para las columnas. En el fichero, cada fila de W será un registro. En efecto: obsérvese que la instrucción 9060 imprime un solo valor de W (el de la fila J y columna K), pero como la instrucción termina con un punto y coma, ese valor no se separa del siguiente por un paso de línea (es decir, de registro). En cambio, una vez que se ha ejecutado por completo el bucle interior (es decir, una vez que se ha escrito una fila de W en el fichero), la instrucción 9080 escribe en éste una cadena vacía (no escribe nada), pero como esta instrucción PRINT no termina en punto y coma, se añade automáticamente un salto de línea (un paso al registro siguiente del fichero).

4. Finalmente, una vez terminados los dos bucles (cuando W se ha escrito por completo), la instrucción 9100 cierra el fichero. Este constará entonces de cuatro registros. El primero contiene los valores de I y de X$; el segundo, la primera fila de W; el tercero, la segunda; y el cuarto, la última fila de W.

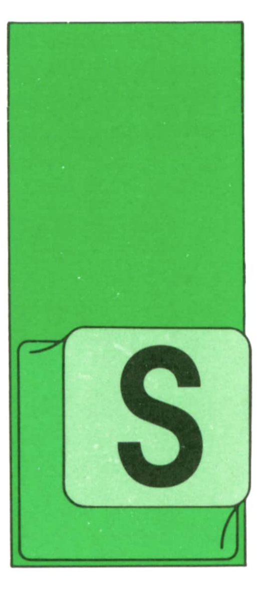
Veamos ahora cómo se pueden leer los datos del fichero creado por el programa anterior desde un programa diferente:

```
10 DIM W(3,5)
. . . . . . .
1000 REM Leer los valores de ciertas
1010 REM variables de un fichero
1020 OPEN "FICHERO" FOR INPUT AS #1
1030 INPUT #1,I,X$
1040 FOR J=1 TO 3
1050 FOR K=1 TO 5
1060 INPUT #1,W(J,K)
1070 NEXT K
1080 NEXT J
1090 CLOSE #1
. . . . . . .
```

La explicación del funcionamiento de este programa es completamente análoga a la del anterior, sustituyendo en cada caso la instrucción INPUT en lugar de PRINT. Asimismo, el fichero debe abrirse para lectura (INPUT).

APLICACIONES

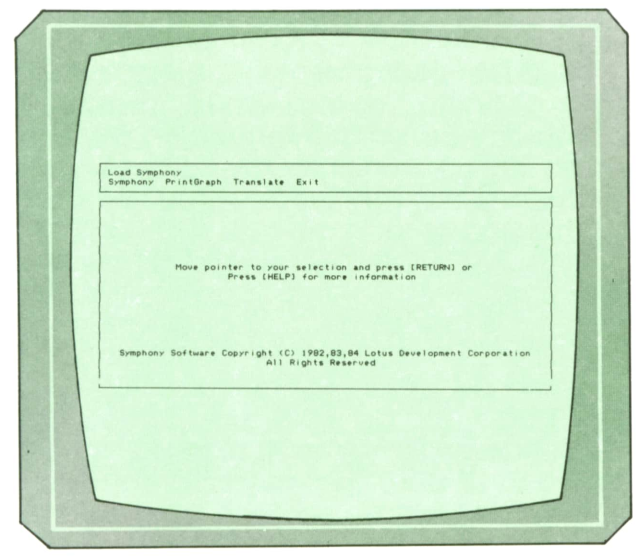
SYMPHONY



YMPHONY es un paquete destinado, en principio, a sustituir al conocidísimo LOTUS 1-2-3, ya que sus funciones constituyen una mejora de este último.

Las funciones que incluye el Symphony son las siguientes:

— Hoja electrónica (SHEET).
— Gestor de base de datos (FORM).
— Generador de gráficos (GRAPH).
— Procesador de textos (DOC).
— Comunicaciones (COMM).




Pantalla de entrada en Symphony.

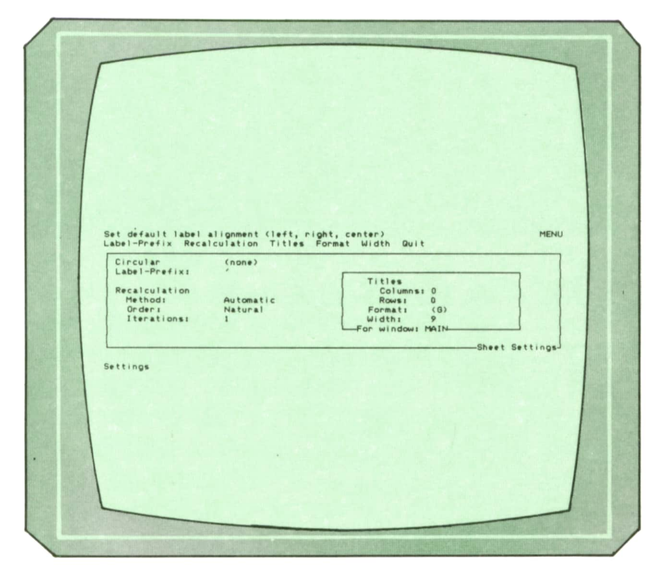
Cada uno de estos módulos puede compartir información con los otros, entrando de esta forma la introducción repetida de los mismos datos en módulos diferentes.

Este programa funciona a base de menús jerarquizados, de forma que al llamar a alguna opción de un menú puede aparecer otro.

Utilizo también las ventanas como medio de presentación de los datos; así, por ejemplo, en una ventana podemos tener la base de datos y en otra el procesador de textos.

Los comandos del Symphony se pueden agrupar en dos tipos básicamente:

Comandos de servicio, encargados de ejecutar órdenes de carácter general y comandos de entorno, utilizables únicamente por cada módulo o enterno particular.




Opciones de celdilla de Symphony.

Dentro de los cómandos de servicios encontramos:

— WINDOW (ventana), opción que permite controlar todos los aspectos de las ventanas, su número, posición, borrado, etc.

— FILE (ficheros), comandos que tienen como misión el manejo de los ficheros externos.

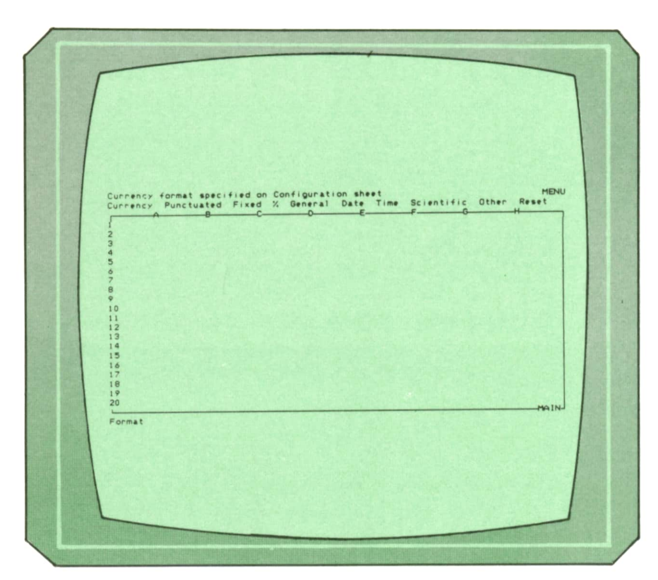
— PRINT (impresión), permite obtener copia en papel de los datos de todos los módulos de Symphony.

— CONFIGURATION adapta el programa al Hardware específico del ordenador empleado.

— APPLICATION (aplicaciones) sirve para ejecutar programas escritos por el usuario y realizar modificaciones en los mismos.

— SETTINGS (características); mediante esta opción se pueden definir características globales de la hoja electrónica.

— NEW (borrado), comando que permite borrar el contenido de una hoja de cálculo.

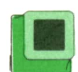




Hoja electrónica de Symphony.

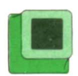
Los comados de entorno incluyen:

— SHEET.
— Hoja electrónica.
— Procesador de datos.
— Base de datos.
— Gráficos.
— Comunicaciones.



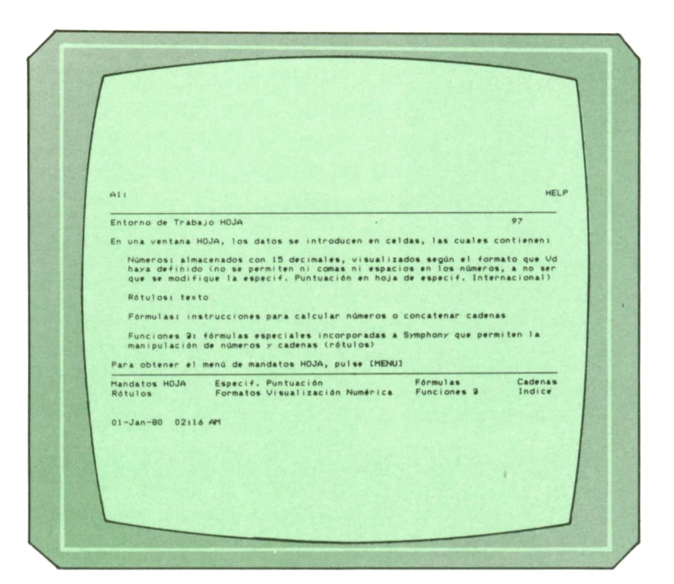
Hoja electrónica

Es el módulo base que constituye el centro del resto de módulos. Permite realizar la opción de una hoja electrónica, tales como: funciones matemáticas, ordenación, previsiones, etc. Soporta hasta 8192 filas y 256 columnas, condicionando el número utilizable de ella para la capacidad de la memoria del ordenador.



Procesador de textos

Con esta opción se pueden crear y modificar documentos y textos de todo tipo, los comandos aparecen al pulsar la tecla F10 encontrándose entre ellos: comandos para borrar, mover, buscar o reemplazar palabras o párrafos. Asimismo, opciones de alineación del margen derecho, y de efectuar con ficheros de la base de datos.





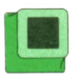
Ayuda de Symphony.



Base de datos

La base de datos es una aplicación muy sencilla, que permite introducir, cambiar y borrar la información almacenada en un conjunto de ficheros. Para manejar esta aplicación se puede crear tanto una ventana del tipo SHEET o FROM, ya que Symphony representa los campos de información en grupos que corresponden a las celdas de la hoja electrónica.

Cada fichero puede almacenar un máximo de 8.000 registros con 250 campos cada uno de ellos, capacidad que depende de la memoria externa. Incluye comandos para formatear la entrada y salida de datos, realizar operaciones con ellos y clasificar varios ficheros utilizando como clave cualquier campo de registro.



Gráficos

Symphony puede crear seis tipos diferentes de gráficos: de coordenadas XY, de líneas, de barras, de barras compuestas, de tarta y gráficos con hasta cuatro rangos (high-low-close-open).

El menú de los gráficos ofrece únicamente cuatro opciones (ATTACH, IMAGE-SAVE y dos SETTINGS). La novedad que in-

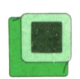
cluye Symphony es la posibilidad de mezclar textos y gráficos en la pantalla simultáneamente, ya que el resto de las opciones (sombreado, anchura, símbolos...) son muy similares a las del resto de los programas.



Comunicaciones

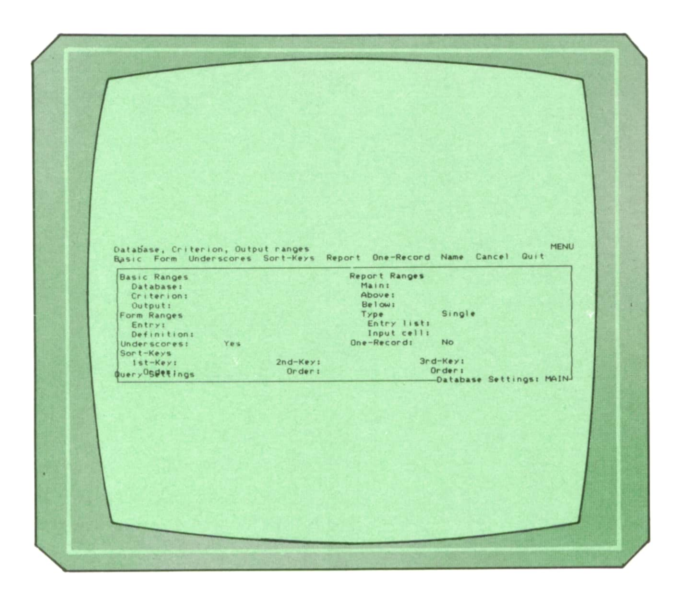
La ventana de trabajo para las comunicaciones permite al usuario de Symphony comunicarse con otros ordenadores, enviar y recibir mensajes, ficheros y archivos de trabajo de la base de datos, hoja electrónica o tratamiento de textos. Soporta comunicaciones asíncronas, utilizando un módem o acoplador acústico, con una velocidad de transmisión entre los 300 y 9.600 baudios.

Symphony permite emular la mayoría de las terminales ANSI-estándar, incluida la emulación parcial del VT-100 de digital. El usuario puede especificar todos los parámetros en la comunicación: velocidad, paridad, longitud de palabras y bits de parada, módulos dúplex, half o full.



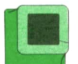
Lenguaje de comandos

Symphony permite al usuario ejecutar una serie de procesos de forma automática, para lo que incluye un conjunto de ochenta comandos con funciones avanzadas, como bucles FOR-NEXT o de bifurcación IF-THEN. De esta forma, se puede crear sus propios comandos o menús como los que utiliza Symphony.





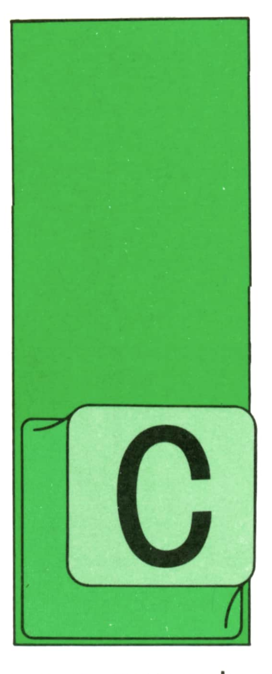
Base de datos de Symphony.



Documentación

Symphony es un programa muy bien presentado y documentado, tanto en manuales como en pantalla: el único inconveniente es que la información no esté en castellano en su totalidad. Symphony se presenta en un estuche de plástico con seis disquetes y cinco manuales. Los disquetes contienen, además del programa Symphony, programas de ayuda (help y tutor) para dar salida a los gráficos por impresoras o plotter. Los cinco manuales son uno de introducción, otro para aprender a manejar el programa, un manual de referencia rápida y un cuadernillo con un glosario de términos informáticos, la estructura de los comandos y los mensajes de error.

PASCAL



Un eliminador de comentarios

COMO ya se ha comentado en otras ocasiones, conviene escribir los programas con abundantes comentarios para así facilitar su comprensión o modificación en un momento posterior, sea por su propio autor o por otra persona.

Sin embargo, los comentarios ocupan espacio, por lo que, si nuestro compilador tuviese un límite en cuanto al tamaño del texto fuente a procesar, podría no ser posible compilar un programa dado debido precisamente a los comentarios. También podría haber, por ejemplo, ocasiones en que los comentarios limitasen bastante la cantidad de programas transportables en un único diskette.

Por todo ello, vamos a realizar un programa capaz de eliminar comentarios de un texto fuente de PASCAL, sin por ello afectar al resultado que se obtenga tras la compilación.

Si se fuese analizando carácter a carácter el texto de un programa PASCAL, en cada momento se estaría en alguna de las siguientes situaciones:

1. en medio de un comentario,
2. en medio de un texto, o
3. en una situación distinta de las dos anteriores.

Para eliminar los comentarios habría que ir analizando uno a uno los caracteres del texto del programa original y observar en qué situación se está en cada momento; cada carácter recogido se añadiría al nuevo programa en fase de formación, excepto cuando se estuviese en medio de un comentario, en cuyo caso habría que limitarse a recorrer el texto hasta cambiar de situación.

Al comenzar el recorrido del texto se estaría, como es lógico, en la situación 3. Sólo se cambiaría de situación cuando se llegase a un apóstrofo (en cuyo caso se pasaría a estar en medio de un texto), a una llave izquierda, o a un par paréntesis izquierdo-asterisco (casos ambos en los que se pasaría a estar en medio de un comentario). Al encontrarse un paréntesis izquierdo, antes de añadirlo al nuevo programa habría que esperar a analizar la siguiente letra para ver si era el comienzo de un comentario y, por tanto, habría que guardar en todo momento la letra anterior para añadirla en caso de que, siendo un paréntesis izquierdo, se comprobase que no iba seguido de un asterisco.

Cuando se está en medio de un texto, la única forma de cambiar de situación es encontrar un nuevo apóstrofo, caso en el que pasaría a la situación 3. La presencia de { o de (* en medio de un texto no tiene ningún significado especial, son simplemente unos caracteres más del texto. Mención aparte merece la presencia de dos apóstrofos seguidos; como constituyen la representación del carácter «apóstrofo» que se utiliza para diferenciarlo del que marca el final del texto, no habría que cambiar, en principio, de situación; sin embargo, resulta más fácil pasar momentáneamente a la situación 3 con el primer apóstrofo del par para regresar con el segundo a la situación original, dado que en ambos casos las acciones a tomar con respecto al nuevo programa son idénticas.

Cuando se está en medio de un comentario, y de manera similar a como sucede cuando se está en medio de un texto, el único cambio posible es el paso a la situación 3 al encontrarse una llave derecha o un par asterisco-paréntesis derecho; la presencia de un apóstrofo no significa nada dentro de un comentario.

Para representar la situación de cada momento, utilizaremos un tipo definido por enumeración. Como es habitual, acudiremos a las convenciones del Turbo Pascal para el manejo de los ficheros:

```
Program QuitaComentarios;
{-------------------------------------------------}
{ Elimina los comentarios de un programa Pascal. }
{-------------------------------------------------}

var
 Estado : (Comentario, Texto, Otro);
 Letra,
 Previa : char;
 Viejo,
 Nuevo  : text;

(****************************************************************)
procedure  AbreFichero (var F: text);
 {------------------------------------------}
 { Lee de teclado el nombre de un fichero y  }
 { lo deja preparado para reset o rewrite    }
 {------------------------------------------}
  var Nombre : array [1..50] of char;
 begin
  readln (Nombre);
  assign (F, Nombre)
 end;

(****************************************************************)
begin
Estado := Otro;
Previa := chr(0);

write ('Nombre del fichero original:   ');
AbreFichero (Viejo);
reset (Viejo);

write ('Nombre del fichero de destino: ');
AbreFichero (Nuevo);
rewrite (Nuevo);

while not eof (Viejo) do
 begin
  read (Viejo, Letra);

  case Estado of
   Texto:
     begin
      write (Nuevo, Letra);
      if Letra = '''' then Estado:= Otro
     end;

   Comentario:
     if (Previa = '*') and (Letra = ')') or (Letra = '}') then
              Estado:= Otro;

   Otro:
     case Letra of
       '{': Estado:= Comentario;

       '(': (* nada hasta ver el siguiente carácter *);
       '*': if Previa = '(' then Estado:= Comentario
                            else write (Nuevo, Letra);
       else
         begin
          if Previa = '(' then write (Nuevo, Previa);
          write (Nuevo, Letra);
          if Letra = '''' then Estado:= Texto
         end
       end { de case Letra }

   end; { de case Estado }

  Previa:= Letra
 end; { de while }

close (Viejo);
close (Nuevo)
end.
```

Como el valor de Previa está indefinido al comenzar el programa, se le asigna uno que no pueda figurar en el programa, como chr (0), para evitar que, por casualidad, se produzca algún error.

Algunos compiladores, como el Turbo Pascal, sólo dan por terminado un comentario al encontrar el delimitador complementario a aquél que lo inició, es decir, al encontrar una llave derecha si se empezó con una llave izquierda y análogamente con los paréntesis y asteriscos. Por ello, habría que diferenciar las dos formas posibles de estar dentro de un comentario, con lo que el programa adecuado sería:

```
Program QuitaComentarios;
{------------------------------------------------}
{ Elimina los comentarios de un programa Pascal. }
{------------------------------------------------}

var
 Estado : (Comentario_1, Comentario_2, Texto, Otro);
 Letra,
 Previa : char;
 Viejo,
 Nuevo  : text;

(****************************************************************)
procedure  AbreFichero (var F: text);
 {-------------------------------------------}
 { Lee de teclado el nombre de un fichero y  }
 { lo deja preparado para reset o rewrite    }
 {-------------------------------------------}
  var Nombre : array [1..50] of char;
 begin
  readln (Nombre);
  assign (F, Nombre)
 end;

(****************************************************************)
begin
Estado := Otro;
Previa := chr(0);

write ('Nombre del fichero original:    ');
AbreFichero (Viejo);
reset (Viejo);

write ('Nombre del fichero de destino: ');
AbreFichero (Nuevo);
rewrite (Nuevo);

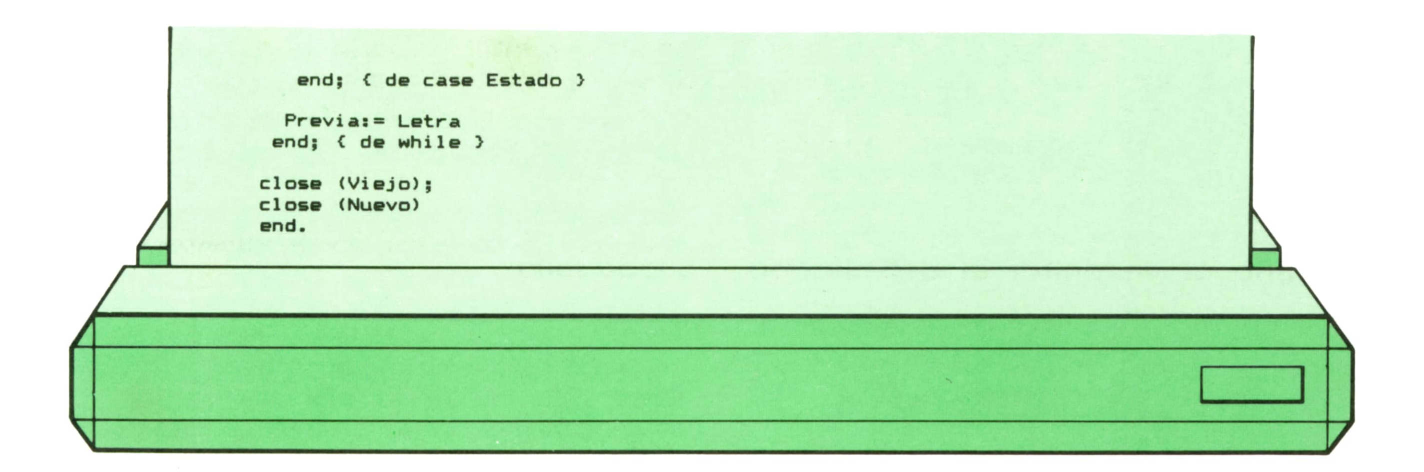
while not eof (Viejo) do
 begin
  read (Viejo, Letra);

  case Estado of
   Texto:
     begin
      write (Nuevo, Letra);
      if Letra = '''' then Estado:= Otro
     end;

   Comentario_1:
     if Letra = '}' then Estado:= Otro;

   Comentario_2:
     if (Previa = '*') and (Letra = ')') then Estado:= Otro;

   Otro:
     case Letra of
       '{': Estado:= Comentario_1;
       '(': (* nada hasta ver el siguiente carácter *);
       '*': if Previa = '(' then Estado:= Comentario_2
                            else write (Nuevo, Letra);
       else
         begin
          if Previa = '(' then write (Nuevo, Previa);
          write (Nuevo, Letra);
          if Letra = '''' then Estado:= Texto
         end
       end { de case Letra }
```



```
    end; { de case Estado }

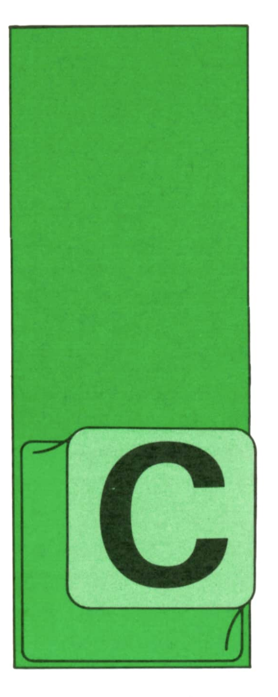
   Previa:= Letra
  end; { de while }

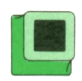
 close (Viejo);
 close (Nuevo)
 end.
```

Con el Turbo Pascal, las denominadas «directivas» de compilación (instrucciones para el compilador que le indican cómo tratar determinados aspectos del programa) figuran como comentarios especiales que se caracterizan por tener el carácter $ inmediatamente a continuación de la marca de comienzo del comentario. Como ejercicio, el lector podría modificar el último programa para que no eliminase esos comentarios; en principio, basta con no dar por comenzado un comentario hasta que no se haya comprobado qué carácter viene a continuación de una llave izquierda o de un conjunto paréntesis izquierdo-asterisco, para lo que habría que guardar no sólo la letra anterior, sino la anterior a ésta.

OTROS LENGUAJES

LISP (II)





El lenguaje LISP (continuación)

COMO dijimos en el capítulo anterior, el lengauje LISP se definía como un lenguaje funcional, simbólico, recursivo, lógico, para el procesado de listas.

Ya tratamos de los tres primeros conceptos, hablemos ahora de los restantes.

La palabra lógico encierra mucho más que el hecho de que la arquitectura de los programas se atenga a una lógica, como en todos los lenguajes. Cuando se diseñó el LISP se pretendió que la lógica fuese una de las esencias del mismo.

Cuando decíamos que en LISP no había ningún dato «predefinido», estábamos diciendo una pequeña mentira (o una media verdad).

En realidad, sí existen tres valores predefinidos y átomos que no pueden contener nada.

Hablemos primero del contenido de las «variables».

Como habíamos dicho, un átomo podía contener otros hasta el infinito, pero hay una excepción. Si el átomo representa un número (por ejemplo, «123»), podríamos decir que su contenido es «la propia esencia del número», la cantidad, y no puede contener «nada más». Sería absurdo pretender igualar 2 a 3 (donde decimos átomo podemos decir también lista).

Por tanto, hay un tipo privilegiado de átomos, los números. Parece razonable esta restricción hecha por los creadores para evitar la hecatombe lógica.

Hablemos ahora de esos tres valores predefinidos, que tiene mucho que ver con la lógica «matemática». Estos son los valores de verdadero, falso e inexistente, representados en LISP por T, F y NIL (del inglés *true*, false y NIL viene de nihil → latín).

Todos los que tengan alguna idea sobre lógica sabrán lo que representan los dos primeros, pero el tercero encierra «algo más». En principio podríamos decir que NIL es otra forma de representar F, pero se utiliza también para otras «cosas» que no estarían demasiado bien representadas con F. Por ejemplo, la variable que está al final de una rama de una estructura arborescente de variables contiene NIL (podríamos hablar más del asunto, pero se alargaría demasiado esta breve introducción al lenguaje).

Además de estos valores predefinidos, muchas funciones estándar de LISP manipulan los datos lógicos, y por qué no decirlo, todo lenguaje de inteligencia artificial debe tener a la lógica muy presente.

Para tranquilizar al lector diré que existen en LISP funciones condicionales, de las que hablaremos más tarde, y que permiten la plena utilización de la lógica en las tareas clásicas de decisión en los programas.

El último concepto que define al LISP es su propio origen, ya que el lenguaje fue diseñado en principio para procesado de listas provenientes del lenguaje natural, aunque luego el concepto de lista de datos se utilizase de una forma mucho más amplia, potente y creativa.

Ahora que ya tenemos una leve idea de a qué nos enfrentamos, hablemos un poco de la sintaxis del lenguaje LISP.

En LISP una función con variables o parámetros se escribe así:

```
(Nombre_de_la_función Parámetro_1 ... Parametro_n)
```

Por ejemplo (PLUS X Y), sería una función que nos devolvería el valor de la suma de X e Y.

La forma de una función es por una afortunada casualidad la misma que tiene una lista. Estas son átomos separados por comas. Así podemos decir que una función es una lista cuyo primer átomo es

el nombre de la misma, y los siguientes son los parámetros que utiliza.

Con esto se acentúa aún más esa regularidad del LISP de la que ya hemos hablado.

Pongamos un ejemplo más complicado. Supongamos que queremos calcular en modo directo, utilizando el intérprete de LISP, la función sen (log (x)), es decir, el seno del logaritmo de un valor *x*. La forma de transcribirlo a LISP sería.

(SIN (LOG X)) o (SIN(LOG X)

Los espacios son separadores cuando no van junto a los paréntesis.

Se puede imaginar fácilmente que el número de paréntesis de una expresión de complejidad media se hace enorme. Es ésta una de las características «marchamo» del LISP.

Cuando se empieza a estudiar el lenguaje LISP, se da uno cuenta de que es un lenguaje constructivo. Este concepto subyace a la propia forma de definirse el LISP. El lenguaje es un conjunto de funciones estándar definidas a partir de unas pocas no definidas en LISP y que podríamos denominar «la semilla máquina del lenguaje». A partir de estas funciones que «aparecen» en la máquina, se definen las pertenecientes al lenguaje. Y basándose en ellas, el usuario define a su vez las suyas propias, para utilizar en sus programas y que también son a su vez funciones utilizables en otros programas, y así hasta el infinito.

En todo este proceso hay una función de especial importancia y es la que sirve para definir otras funciones, de la que hablaremos posteriormente.

Hablemos ahora más detalladamente de los datos para, en el próximo capítulo, poder adentrarnos definitivamente en la programación LISP.

Cuando explicamos en qué consistían las listas, no dejamos claro que los miembros de las mismas podían ser otras listas, lo cual ofrece una nueva perspectiva multidimensional de los datos.

No sería, por ejemplo, lo mismo:

(A B C D) que (A (B C) D)

En el primer caso tendríamos una lista de cuatro elementos (átomos u otra cosa, pero todos del mismo rango). Y en el segundo tenemos una lista de tres elementos, el segundo de los cuales es otra lista (en este caso la llamaríamos sublista). Piense el lector en otras posibilidades, y en lo que podrían representar.

Para dejar el tema de los datos en LISP, debemos tratar ahora de un tipo del que no hemos hablado para simplificar la explicación y al que se suele llamar par apuntado *(dotted pair).*

Un par apuntado es un tipo de dato que está formado exclusivamente por dos datos simples (ni más ni menos) y que se representa en LISP así:

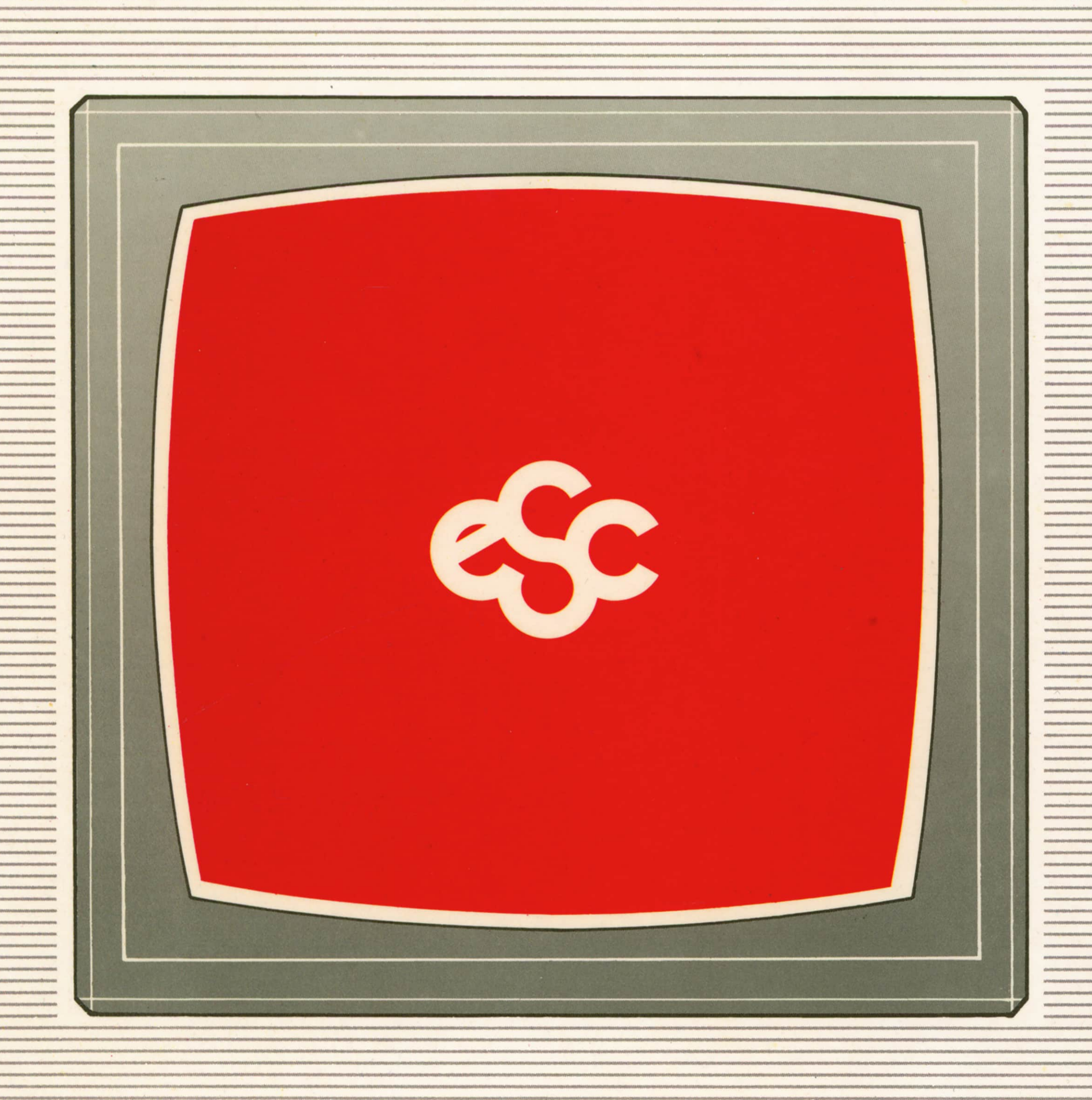
(Dato_1 . Dato_2)

Es decir, dos datos separados por un punto, por ejemplo (X.Y) sería un «dotted pair».

¿Qué representa? Podríamos decir que el primer dato (o variable) contiene un puntero (o apuntador) al segundo. El contenido del segundo puede ser cualquier dato LISP, un átomo, una lista u otro par apuntado. Obsérvese la gran similitud con las estructuras de punteros (o apuntadores) del PASCAL.

La enorme ventaja de esta estructura de punteros es que combinada con la extraordinaria libertad de datos-variables que proporciona el LISP y al no haber ningún tipo de restricción para utilizarse combinada y en cualquier lugar, proporciona las máximas posibilidades para la representación de datos que posean algún tipo de encadenamiento.

En el próximo y último capítulo de esta miniserie trataremos ya de una forma lo más práctica posible de lo que se puede hacer con este lenguaje, de sus funciones principales y algún ejemplo significativo.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼